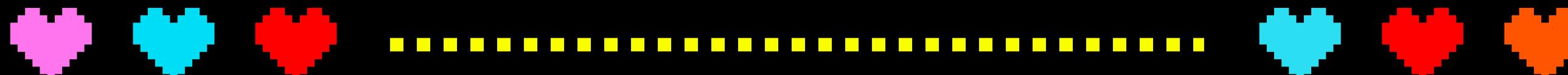


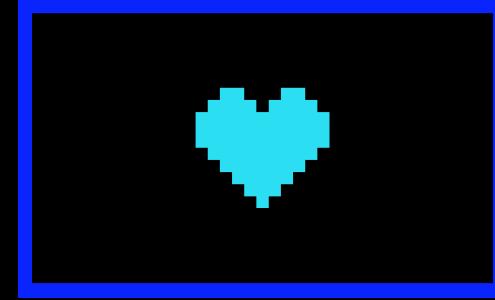
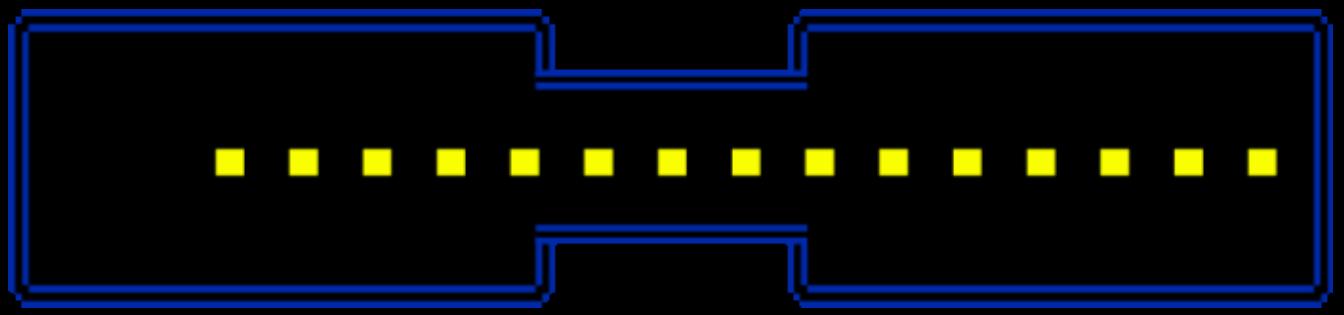
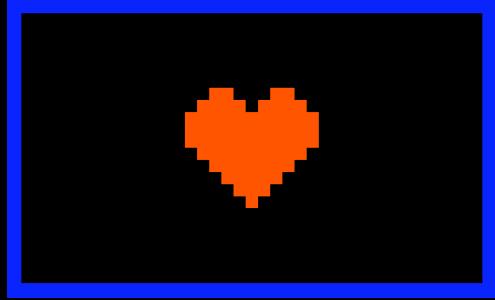
# INTRODUÇÃO A DESENVOLVIMENTO DE JOGOS

COM JÁDER LOUIS

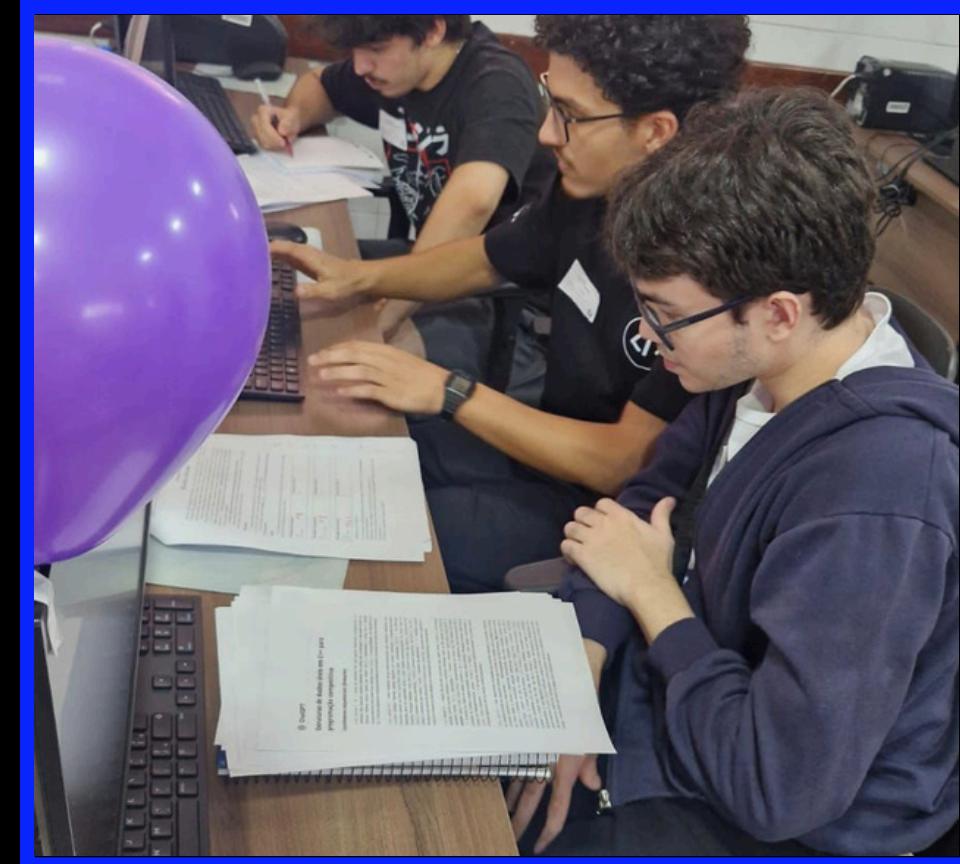


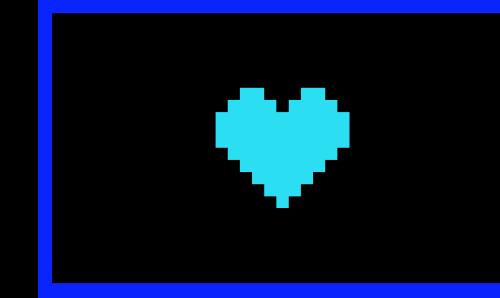
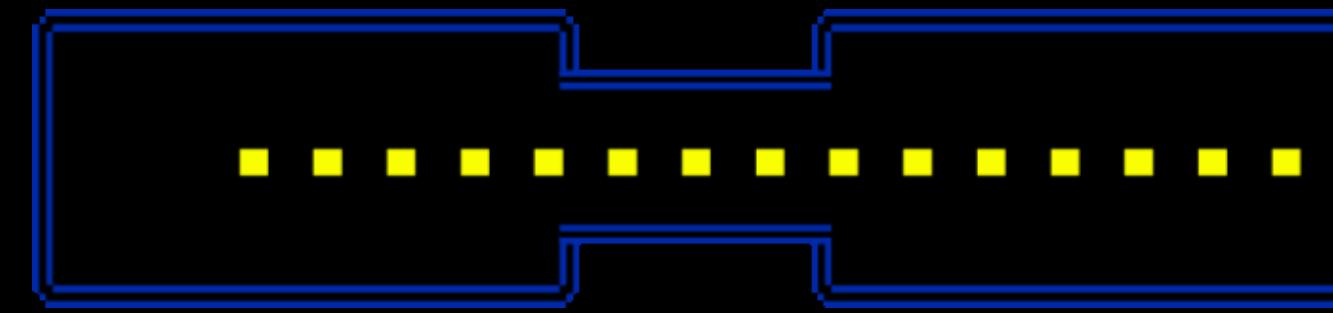
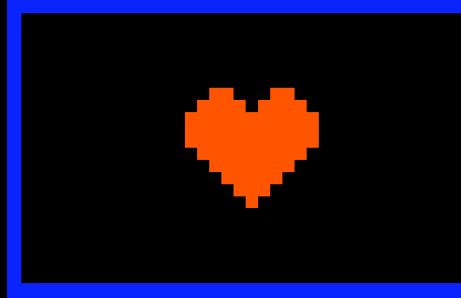
UNIVERSIDADE FEDERAL DE  
RONDÔNIA

DEPARTAMENTO ACADÊMICO  
DE CIÊNCIA DA COMPUTAÇÃO



# BEM VINDOS A VIII SEMANA DA COMPUTAÇÃO!





# QUEM SOU EU?

ELETROTECNICO



8º PERÍODO - UNIR

DESENVOLVEDOR GEODESPACIAL - COGEO / SEDAM

INTEGRANTE DO GRUPO DE PESQUISA DE VISÃO  
COMPUTACIONAL - UNIR

PESQUISADOR NA ÁREA DE LLMS, TRANSFORMERS E  
OTIMIZAÇÃO DE GRAFOS

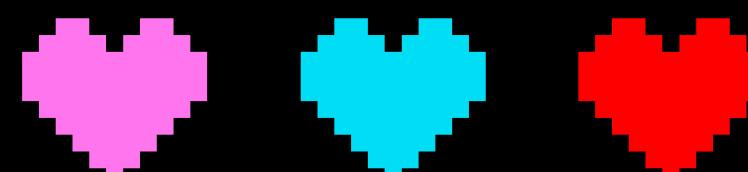
O QUE SERÁ ABORDADO NO CURSO?

OVERVIEW NO CENÁRIO BRASILEIRO  
DE DESENVOLVIMENTO DE JOGOS

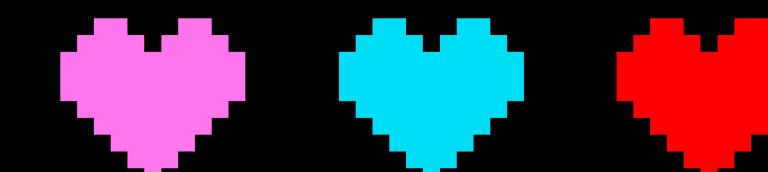
CONCEITOS SOBRE GAME DESIGN

TÉCNICAS DE DESENVOLVIMENTO

DINÂMICAS ENVOLVENDO PRÊMIOS



PARTE I



# OVERVIEW NO CENÁRIO BRASILEIRO DE DESENVOLVIMENTO DE JOGOS



# OVERVIEW NO CENÁRIO BRASILEIRO DE DESENVOLVIMENTO DE JOGOS



JOGOS TAMBÉM SÃO PROJETOS;  
JOGOS TAMBÉM SÃO TRABALHOS.

JOGOS TAMBÉM TEM

**RELEVÂNCIA CIÊNTIFICA**

# OVERVIEW NO CENÁRIO BRASILEIRO DE DESENVOLVIMENTO DE JOGOS

## A Scalable and Production Ready Sky and Atmosphere Rendering Technique

Sébastien Hillaire <sup>1</sup>

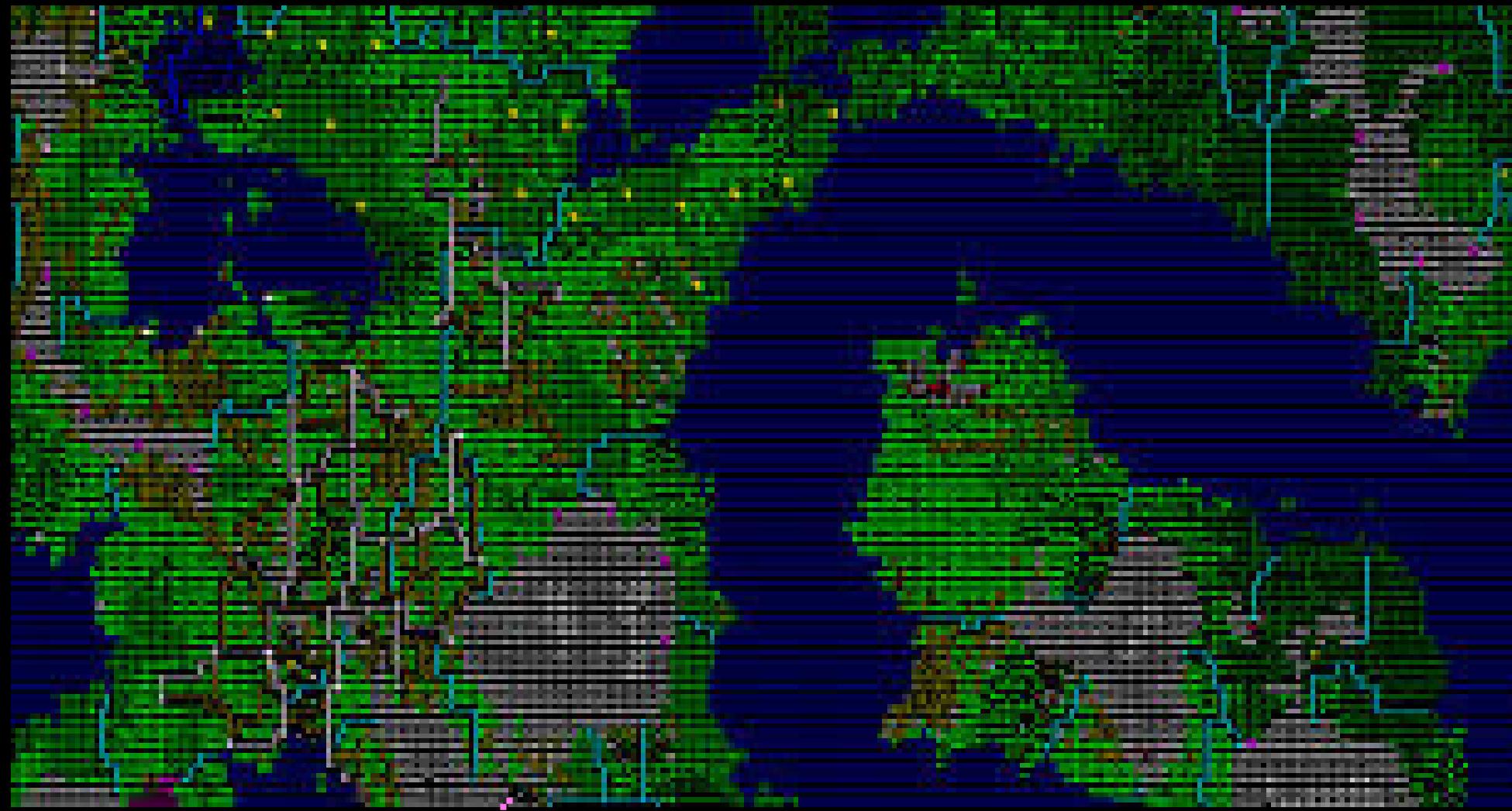
<sup>1</sup>Epic Games, Inc



**Figure 1:** Rendered images of different atmospheric conditions and view points using the method presented in this article. Left to right: ground views of an Earth-like daytime and Mars-like blue sunset, and space views of an Earth-like planet and an artistic vision of a tiny planet.

JOGOS TAMBÉM TEM  
RELEVÂNCIA CIÊNTIFICA

# OVERVIEW NO CENÁRIO BRASILEIRO DE DESENVOLVIMENTO DE JOGOS



700K LINHAS DE CÓDIGO, 20 ANOS DE DESENVOLVIMENTO

JOGOS TAMBÉM TEM

**RELEVÂNCIA CIÊNTIFICA**

# OVERVIEW NO CENÁRIO BRASILEIRO DE DESENVOLVIMENTO DE JOGOS



MAIOR MERCADO DA AMÉRICA LATINA

10º MAIOR DO PLANETA

EM CRESCENTE!!!



# OVERVIEW NO CENÁRIO BRASILEIRO DE DESENVOLVIMENTO DE JOGOS

COMO É O MERCADO PARA  
DESENVOLVEDORES?

PONTOS POSITIVOS

- POUCA CONCORRÊNCIA
- FÁCIL ENTRADA
- BONS SALARIOS
- TRABALHAR COM O QUE GOSTA

PONTOS NEGATIVOS

- INSTABILIDADE
- MUITA PRESSÃO
- PIRATARIA

# OVERVIEW NO CENÁRIO BRASILEIRO DE DESENVOLVIMENTO DE JOGOS

SALÁRIO MÉDIO DE UM  
DESENVOLVEDOR DE JOGOS NO  
BRASIL

R\$ 5.800,00



RANGE DO SALÁRIO

R\$ 2.100,00 - R\$ 35.000,00

FONTES: LINKEDIN, GLASSDOOR, REDDIT

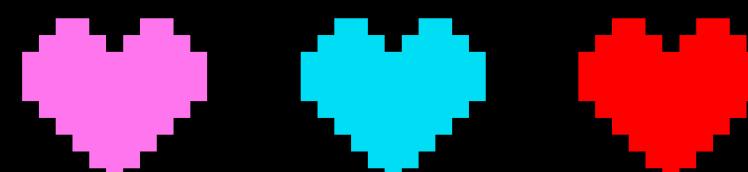
O QUE SERÁ ABORDADO NO CURSO?

OVERVIEW NO CENÁRIO BRASILEIRO  
DE DESENVOLVIMENTO DE JOGOS

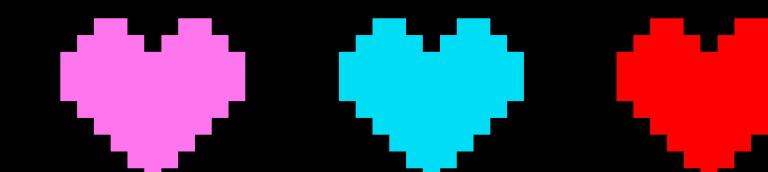
CONCEITOS SOBRE GAME DESIGN

TÉCNICAS DE DESENVOLVIMENTO

DINÂMICAS ENVOLVENDO PRÊMIOS



PARTE 2



MENU

01

07

12



# PILARES DO GAME DESIGN

## MECÂNICAS

AS REGRAS, SISTEMAS E INTERAÇÕES QUE O JOGADOR PODE FAZER.

EX: SUPER MARIO: PULAR, CORRER, COLETAR MOEDAS, PISAR EM INIMIGO

## NARRATIVA

A HISTÓRIA, CONTEXTO E MOTIVAÇÃO QUE DÁ SIGNIFICADO ÀS AÇÕES DO JOGADOR.

EX: HISTÓRIA QUE SURGE DO GAMEPLAY (MINECRAFT, RIMWORLD)

MENU

01

07

12



# PILARES DO GAME DESIGN

## ESTÉTICA

◆ A APRESENTAÇÃO VISUAL E SONORA QUE CRIA A ATMOSFERA E COMUNICA INFORMAÇÕES.

EX: ARTE: SPRITES, MODELOS, UI, ANIMAÇÕES

## TECNOLOGIA

◆ COMO ISSO FUNCIONA?

EX: AS FERRAMENTAS, ENGINES E CÓDIGO QUE TORNAM TUDO POSSÍVEL.

MENU

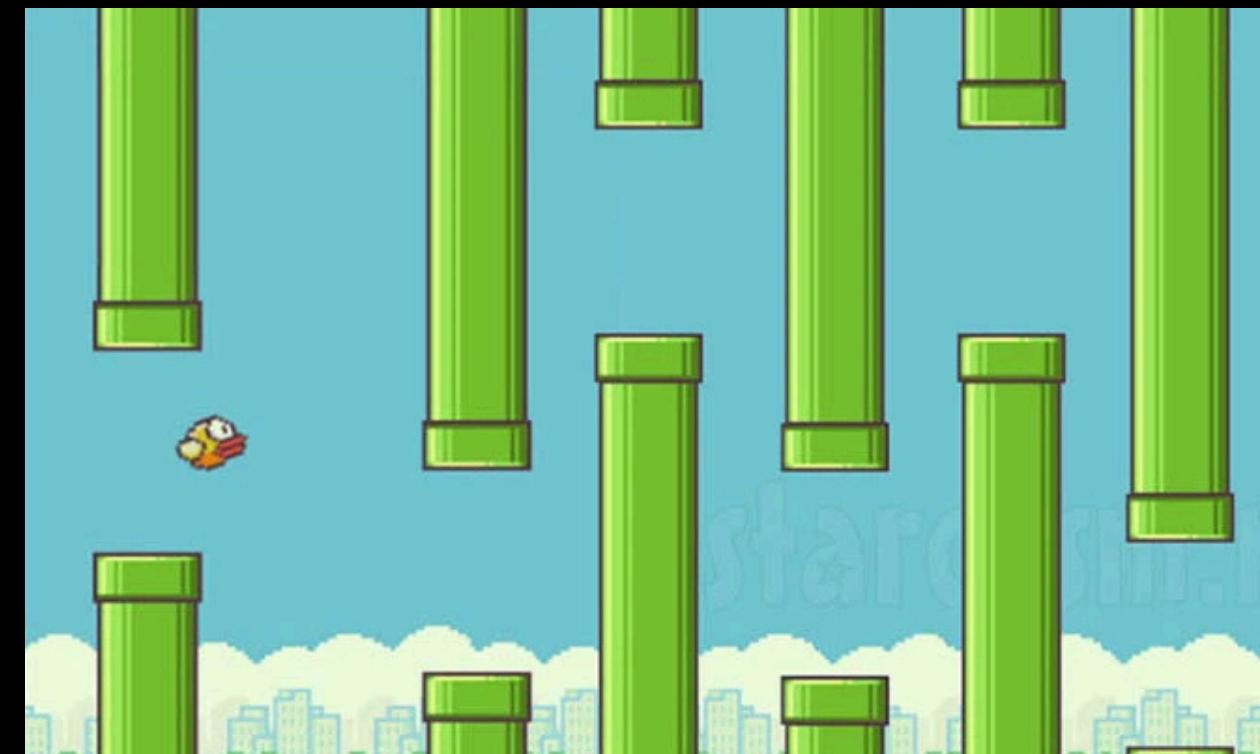
⚔ 01

♦ 07

★ 12



# PILARES DO GAME DESIGN



VALENDO BRINDE!!!

MECÂNICA: ?

NARRATIVA: ?

MENU

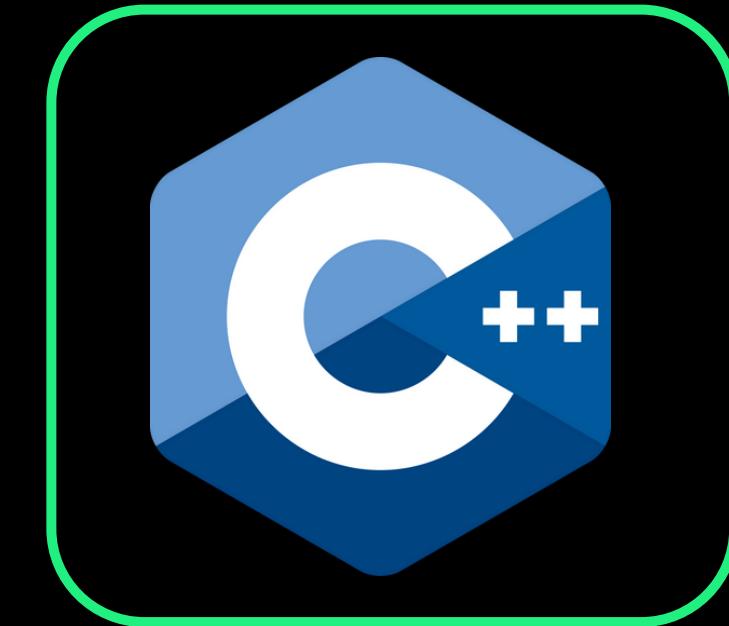
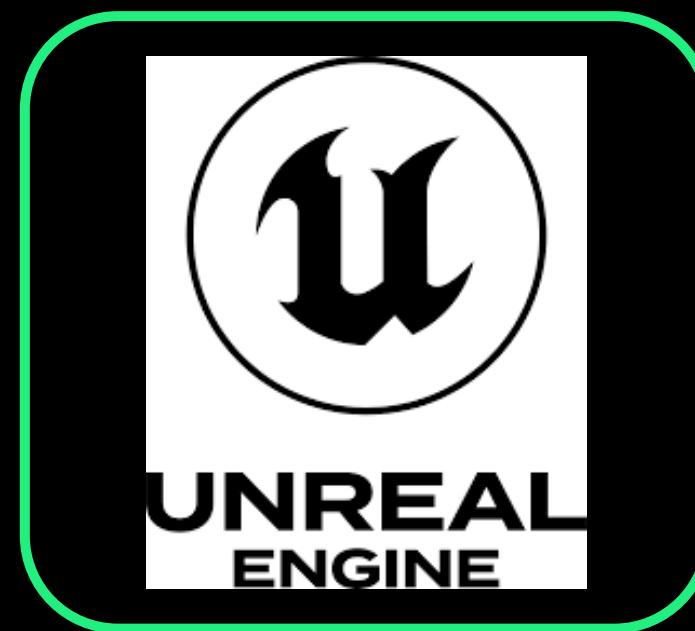
➤ 01

♦ 07

★ 12



# TECNOLOGIA



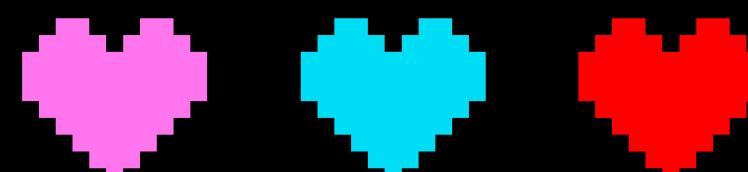
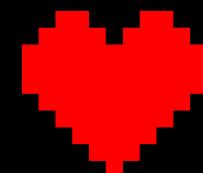
O QUE SERÁ ABORDADO NO CURSO?

OVERVIEW NO CENÁRIO BRASILEIRO  
DE DESENVOLVIMENTO DE JOGOS

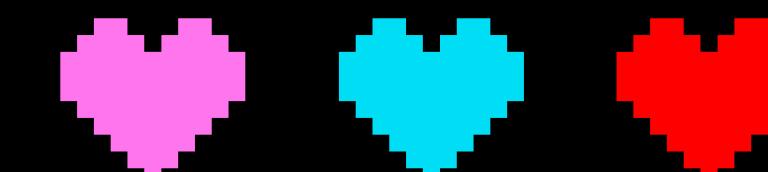
CONCEITOS SOBRE GAME DESIGN

TÉCNICAS DE DESENVOLVIMENTO

DINÂMICAS ENVOLVENDO PRÊMIOS



PARTE 3



MENU

➤ 01

♦ 07

★ 12



# TECNOLOGIA



PYTHON --VERSION

SE TIVER PYTHON, OK!  
PRÓXIMO PASSO!

PIP INSTALL PYGAME

AGORA É ESPERAR BAIXAR!

MENU

➡ 01

♦ 07

★ 12



# TECNOLOGIA

```
import pygame
import sys

# Inicializa o Pygame
pygame.init()

screen = pygame.display.set_mode((400, 300)) # Cria a Tela 400x300
pygame.display.set_caption("Teste Pygame") #Define nome da Janela

running = True # Váriavel de Controle do estado do Jogo
```

CONFIGURAÇÃO INICIAL - 1

# TECNOLOGIA

```
while running:    # Enquanto estiver rodando
    for event in pygame.event.get(): # Escuta por evento
        if event.type == pygame.QUIT: # se tiver algum evento:
            running = False # estado do jogo = falso, fim de jogo!

    # Preenche a tela com cor azul
    screen.fill((0, 100, 200))
    # Atualiza a tela
    pygame.display.flip()
```

## CONFIGURAÇÃO INICIAL - 2

MENU

01

07

12



# TECNOLOGIA

```
pygame.quit() # Encerra Jogo  
sys.exit() # Encerra Janela
```

CONFIGURAÇÃO INICIAL - 3

MENU

➡ 01

♦ 07

★ 12



# TECNOLOGIA

```
import pygame
import sys

# Inicializa o Pygame
pygame.init()

screen = pygame.display.set_mode((400, 300)) # Cria a Tela 400x300
pygame.display.set_caption("Teste Pygame") # Define nome da Janela

running = True # Variável de Controle do estado do Jogo
while running: # Enquanto estiver rodando
    for event in pygame.event.get(): # Escuta por evento
        if event.type == pygame.QUIT: # se tiver algum evento:
            running = False # estado do jogo = falso, fim de jogo!

    # Preenche a tela com cor azul
    screen.fill((0, 100, 200))

    # Atualiza a tela
    pygame.display.flip()

pygame.quit() # Encerra Jogo
sys.exit() # Encerra Janela
```

**MENU**

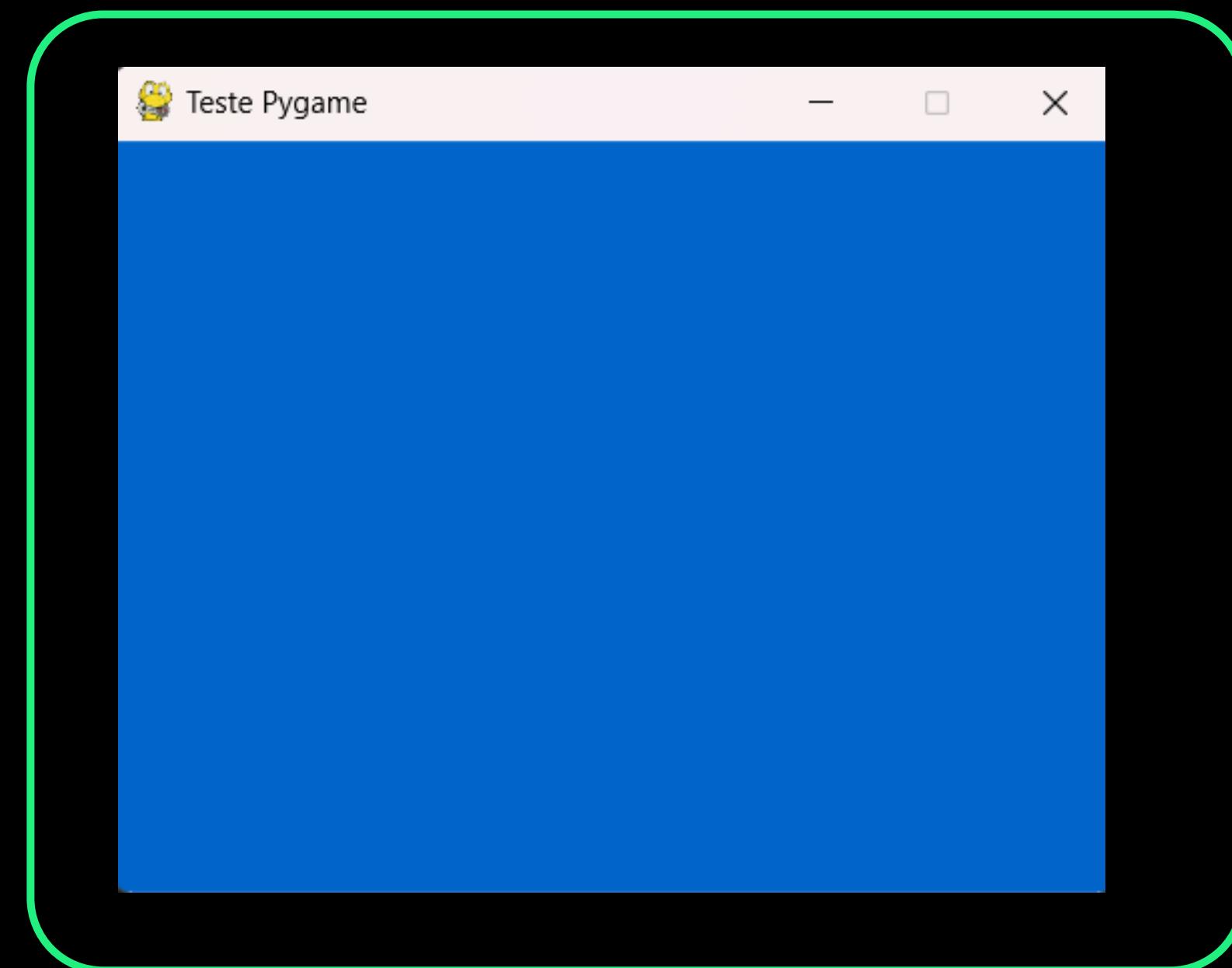
→ 01

◆ 07

★ 12



# RESULTADO:



MENU

01

07

12



# O QUE É FPS?

## IMAGENS POR SEGUNDO!

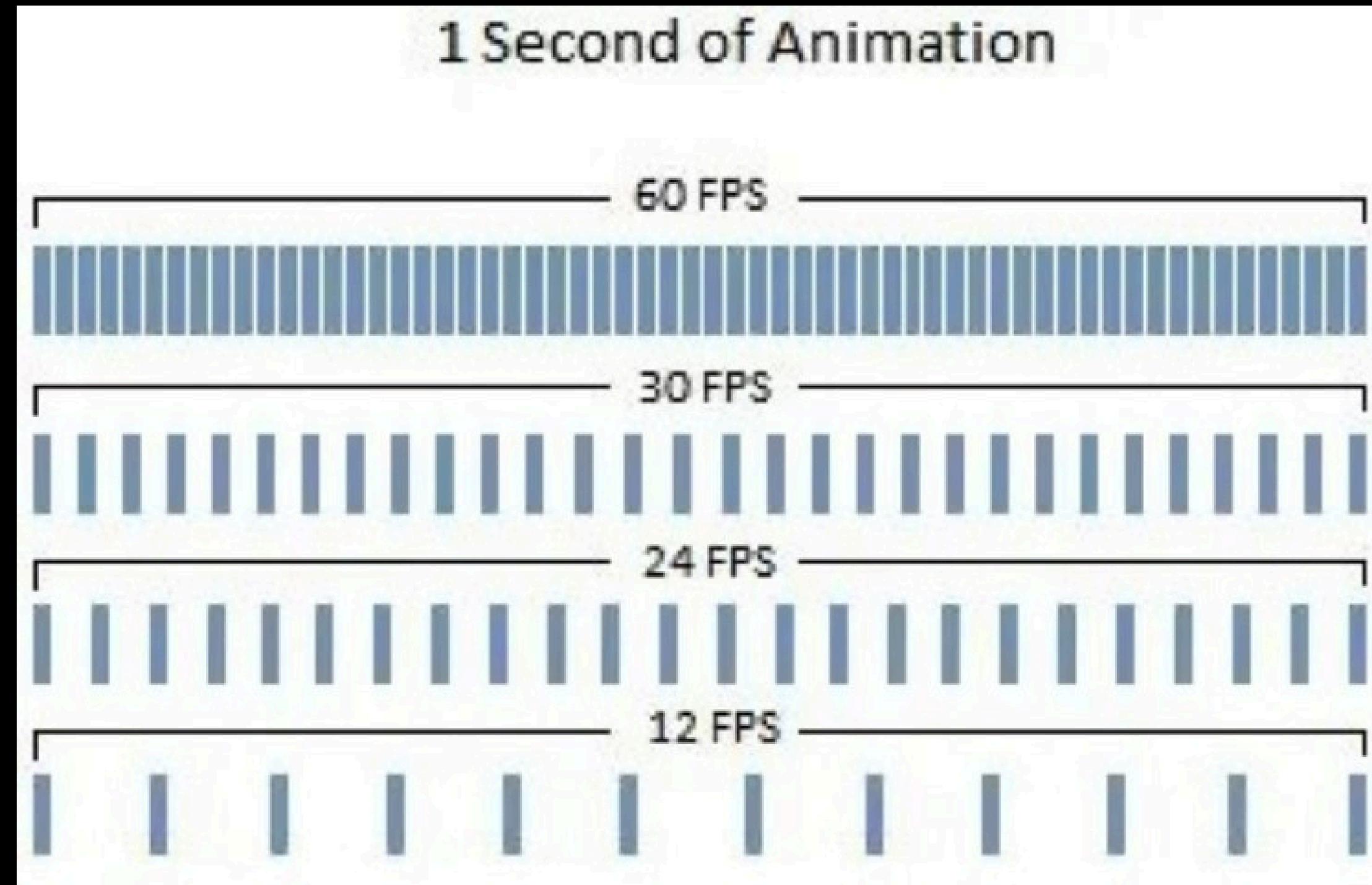
1 Second of Animation

60 FPS

30 FPS

24 FPS

12 FPS



MENU

01

07

12



# O QUE É FPS?

NOSSO JOGO ESTÁ  
RENDERIZANDO A QUASE  
10K FRAMES P/ SEGUNDO

COMO CONSEGUIMOS?

MENU

01

07

12



O QUE É FPS?

NO PYGAME TEMOS:

```
clock = pygame.time.Clock()
```

```
clock.tick(60)
```

CRIAMOS O OBJETO (CLOCK)

SETAMOS O CLOCK PARA 60 FPS

MENU

01

07

12



# DESAFIO!

```
clock = pygame.time.Clock()
```

```
clock.tick(60)
```

DICA: DECLARAÇÃO DE  
VARIÁVEIS NO INÍCIO

DICA: INSERÇÃO DE  
MÉTODOS DENTRO DO LOOP

```
print(clock.get_fps()) # Exibe o FPS atual no console
```

TENTE LIMITAR O FPS DO  
SEU JOGUITINHO!

MENU

01

07

12



# RESULTADO:

60.60606002807617  
60.60606002807617  
60.60606002807617  
60.60606002807617  
60.60606002807617  
60.60606002807617  
60.60606002807617  
60.975608825683594  
60.975608825683594  
60.975608825683594

MENU

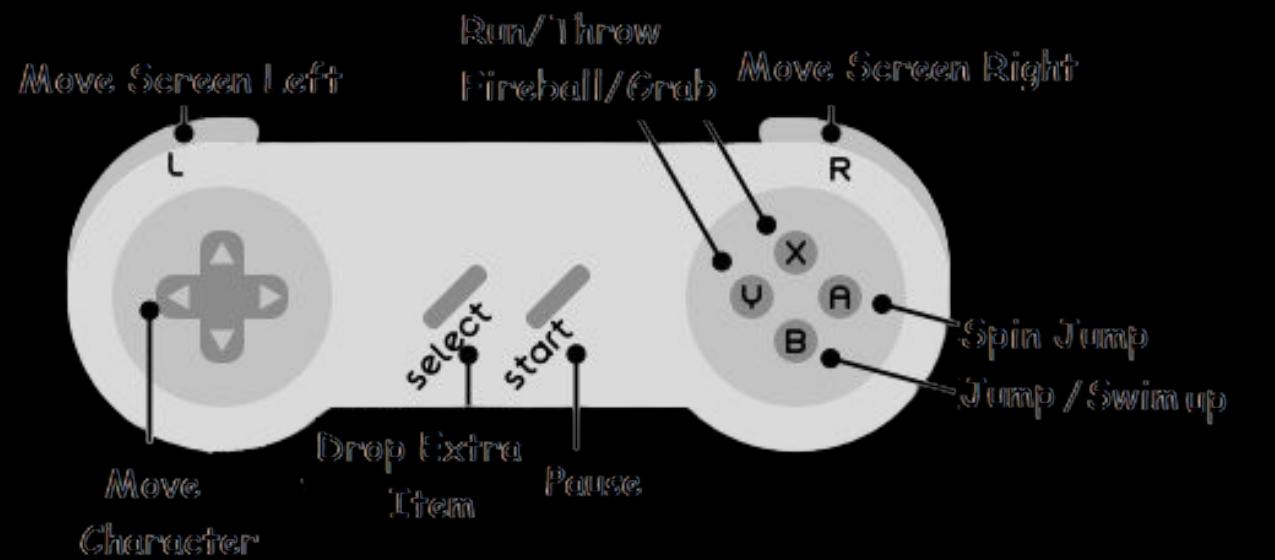
01

07

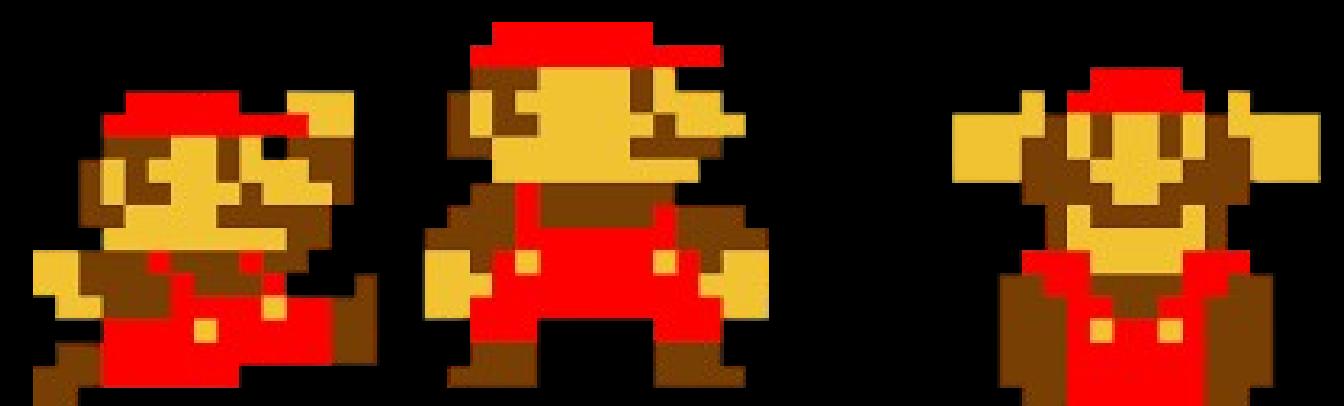
12



# CONTROLES E EVENTOS



PRESSIONAMOS BOTÕES E CONSEGUIMOS  
CONTROLAR O MARIO!



MENU

01

07

12



# CONTROLES

```
event.type == pygame.KEYDOWN
```

CHECA SE O USUARIO ESTÁ  
PRESSIONANDO UMA TECLA

```
if event.key == pygame.K_r:
```

VERIFICA SE A TECLA QUE ESTA SENDO  
PRESSIONADA E A TECLA DO TECLADO "R"  
• TEMOS K DE "KEYBOARD"

MENU

01

07

12



# VAMOS ADICIONAR CONTROLES?

```
# NOVO: Detectar teclas pressionadas
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_r: # Tecla R
        print("R pressionado!")
```

## RESULTADO:

```
R pressionado!
R pressionado!
R pressionado!
R pressionado!
```

MENU

01

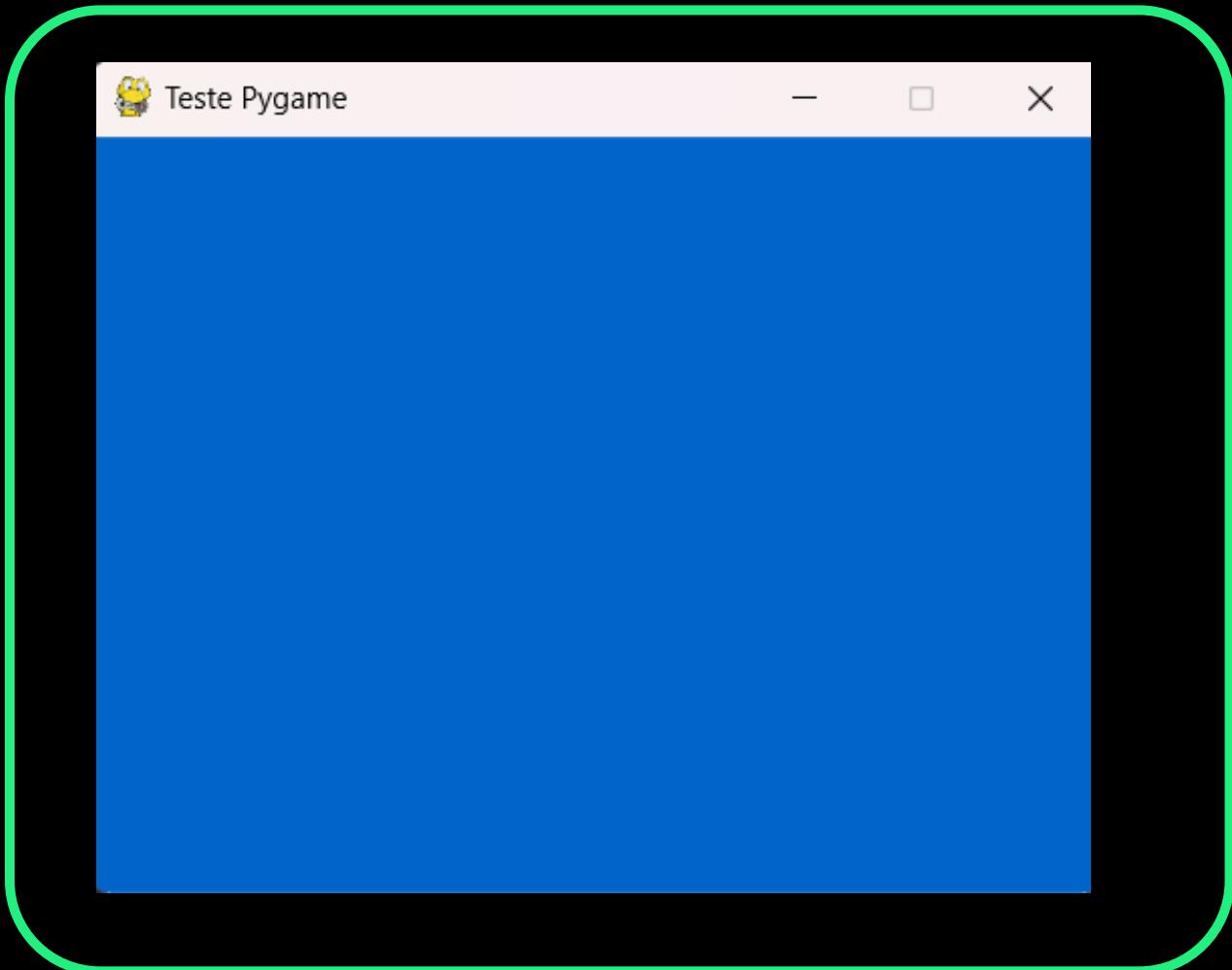
07

12



# REATIVIDADE

```
screen.fill((0, 100, 200))
```



DEFINE A COR DE EXIBIÇÃO  
NO FORMATO RGB

```
SCREEN.FILL( R , G , B )
```

PODEMOS CONTROLAR ISSO  
ATRIBUINDO UMA VARIÁVEL  
A ESTA FUNÇÃO!

```
# Variável guarda o ESTADO atual da cor  
cor_fundo = (0, 100, 200)
```

RENDERIZAMOS NO LOOP

```
screen.fill(cor_fundo)
```

MENU

01

07

12

# DESAFIOS!

```
# Variável guarda o ESTADO atual da cor
cor_fundo = (0, 100, 200)
running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        # Quando apertar uma tecla, MUDA o estado
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_r:
                pass
                #Mude a cor para vermelho
            elif event.key == pygame.K_g:
                pass
                #Mude a cor para verde
            elif event.key == pygame.K_b:
                pass
                #Mude a cor para azul

    # Sempre pinta com a cor atual (fora do loop de eventos)
    screen.fill(cor_fundo)
```

screen.fill((0, 100, 200))

DEFINE A COR DE EXIBIÇÃO  
NO FORMATO RGB

SCREEN.FILL(R, G, B)

PODEMOS CONTROLAR ISSO  
ATRIBUINDO UMA VARIÁVEL  
A ESTA FUNÇÃO!

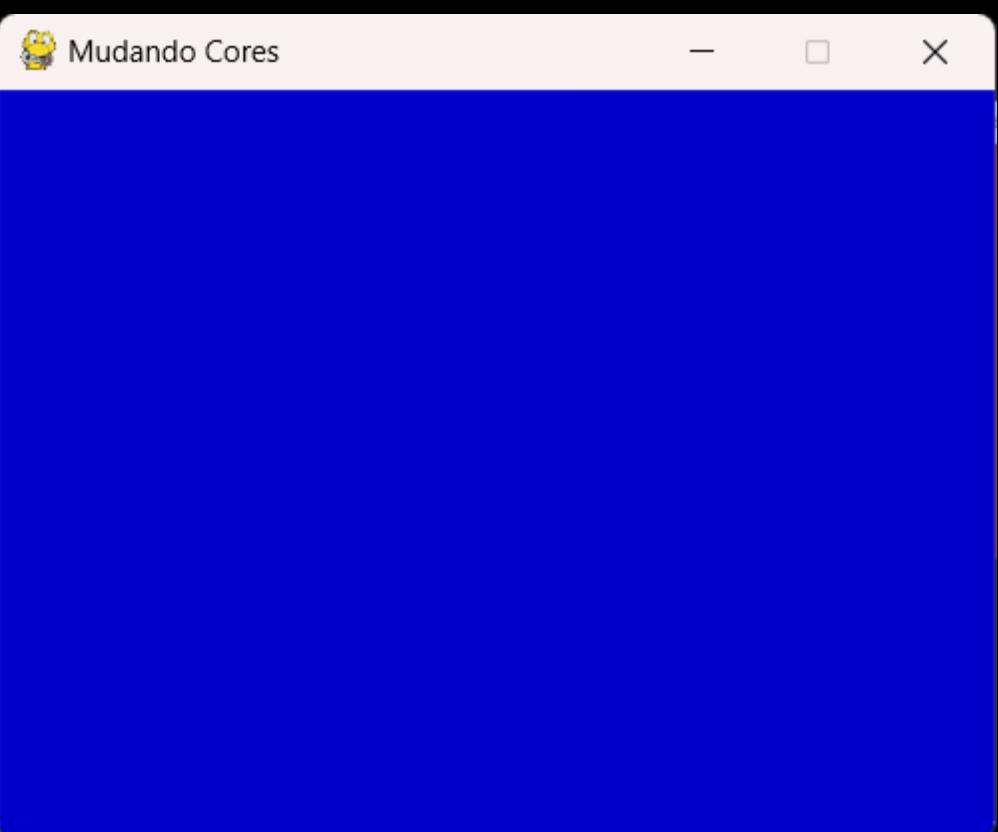
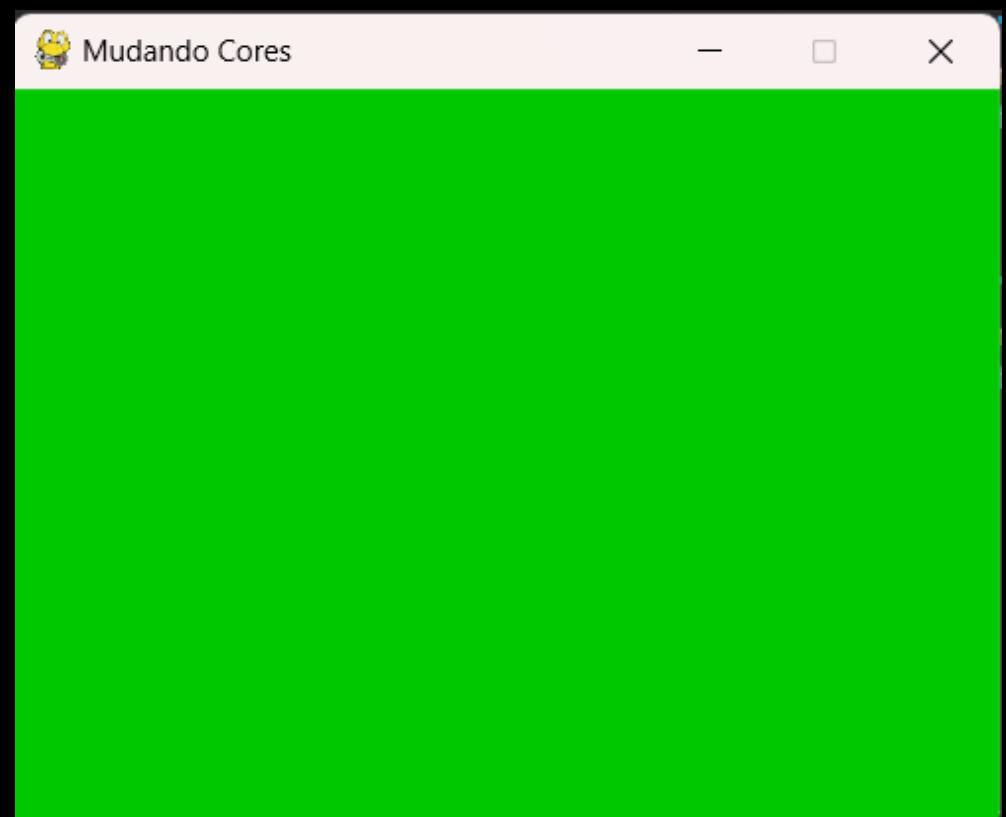
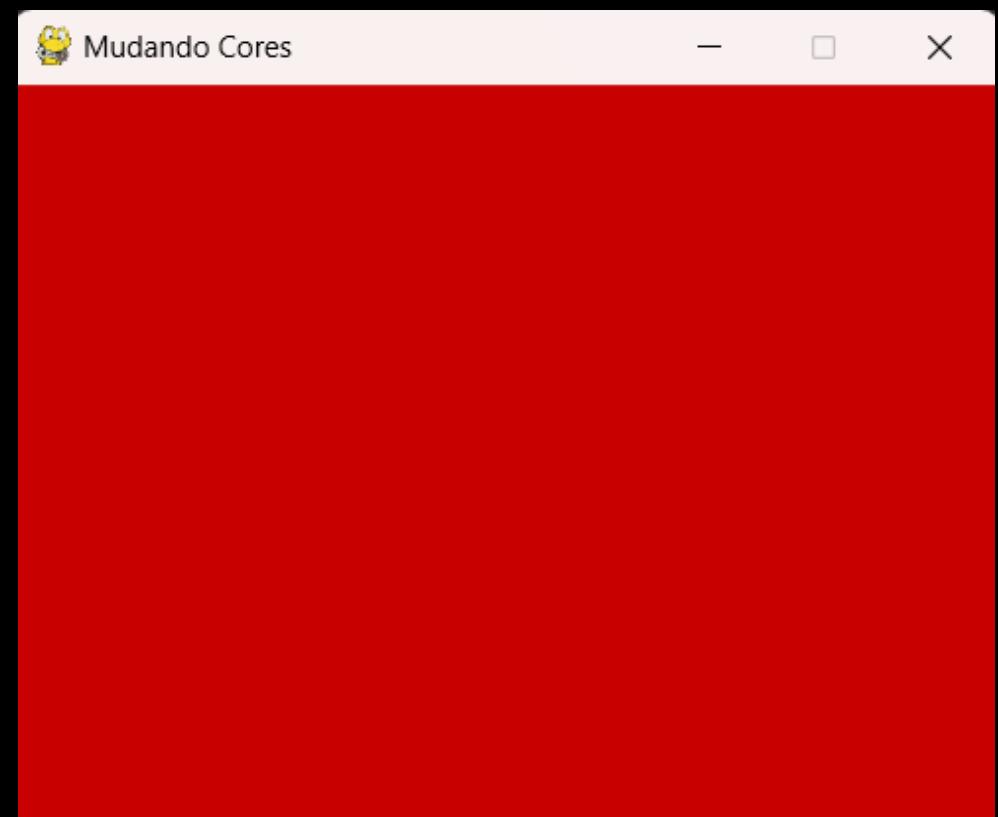
```
# Variável guarda o ESTADO atual da cor
cor_fundo = (0, 100, 200)
```

RENDERIZAMOS NO LOOP

screen.fill(cor\_fundo)

# RESULTADO:

```
while running:  
    clock.tick(60)  
  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            running = False  
  
        # NOVO: Detectar teclas pressionadas  
        if event.type == pygame.KEYDOWN:  
            if event.key == pygame.K_r: # Tecla R  
                cor_fundo = (200, 0, 0) # Vermelho  
            elif event.key == pygame.K_g: # Tecla G  
                cor_fundo = (0, 200, 0) # Verde  
            elif event.key == pygame.K_b: # Tecla B  
                cor_fundo = (0, 0, 200) # Azul  
  
    # NOVO: Usar a variável cor_fundo  
    screen.fill(cor_fundo)  
    pygame.display.flip()
```



MENU

➤ 01

♦ 07

★ 12

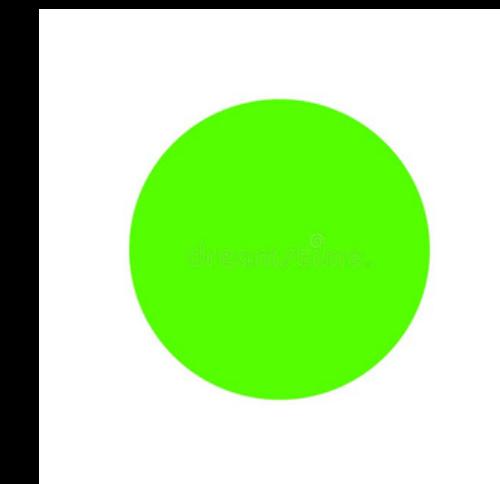
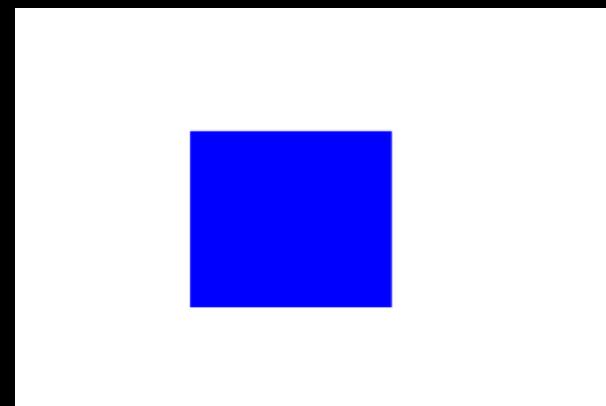


# SPRITES, FORMAS, MODELOS

## FORMAS

SÃO PRIMITIVAS GEOMÉTRICAS USADAS EM JOGOS PARA DIVERSOS PROPÓSITOS:

- COLISÕES: FORMAS SÃO FUNDAMENTAIS PARA DETECÇÃO DE COLISÃO (COLLISION DETECTION).
- RETÂNGULOS, CÍRCULOS, LINHAS, POLÍGONOS E OUTRAS FORMAS GEOMÉTRICAS SIMPLES QUE PODEM SER DESENHADAS DIRETAMENTE SEM USAR SPRITES/TEXTURAS.



MENU

01

07

12



# SPRITES, FORMAS, MODELOS

## SPRITES

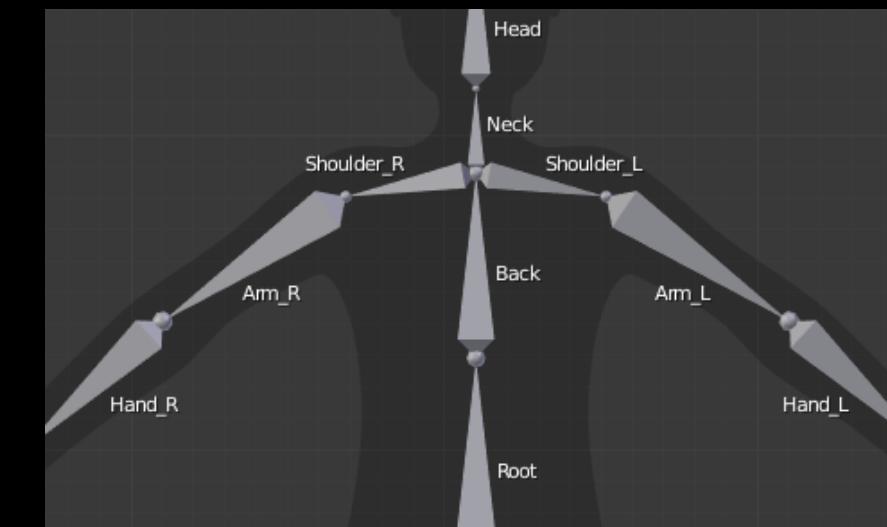
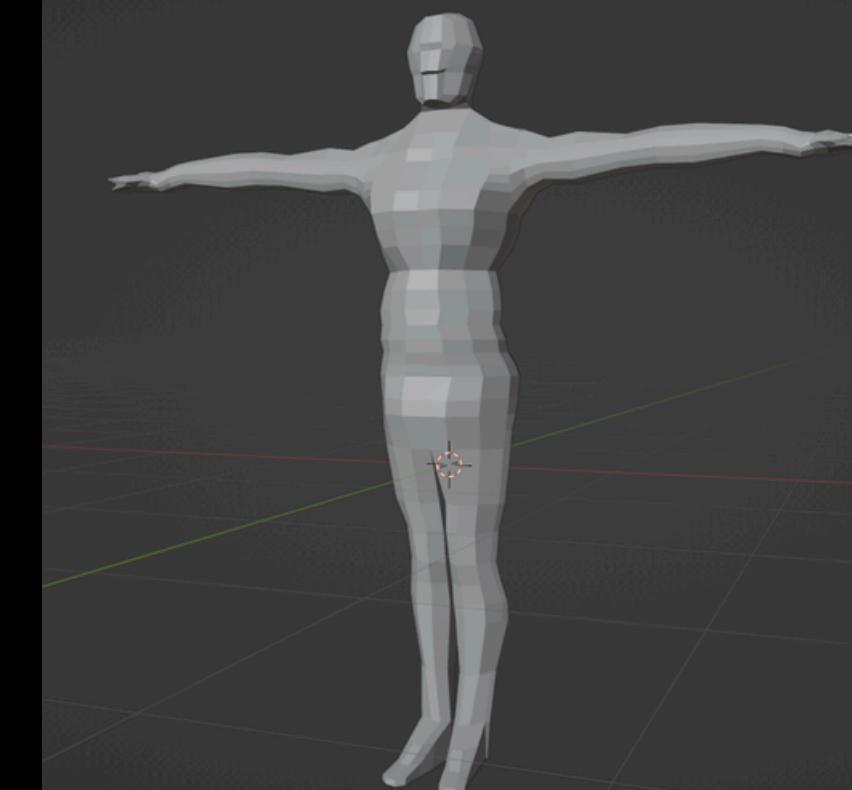
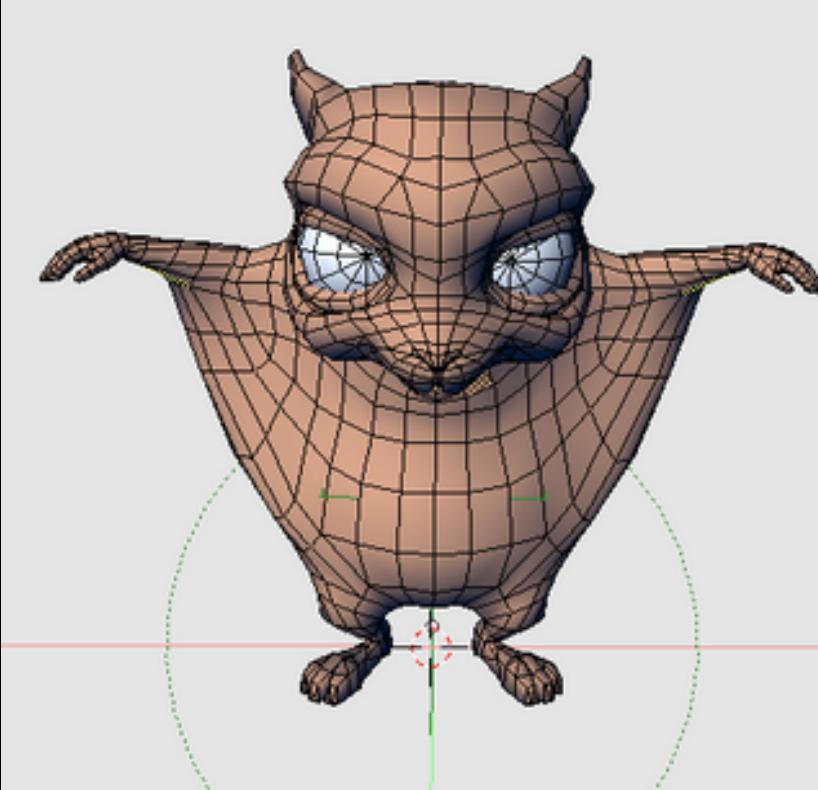
SPRITES SÃO IMAGENS 2D USADAS PARA REPRESENTAR OBJETOS, PERSONAGENS E ELEMENTOS VISUAIS EM JOGOS. ELES SÃO ESSENCIALMENTE BITMAPS OU TEXTURAS PLANAS QUE SÃO DESENHADAS NA TELA.



# SPRITES, FORMAS, MODELOS

## MODELOS

UM MODELO 3D É UMA ESTRUTURA MATEMÁTICA COMPOSTA POR VÉRTICES (PONTOS NO ESPAÇO 3D), ARESTAS E FACES QUE DEFINEM A GEOMETRIA DE UM OBJETO TRIDIMENSIONAL.



MENU

01

07

12



# RENDERIZANDO OBJETOS

NOSSO JOGUINHO ESTÁ  
VAZIO! COMO PODEMOS  
ADICIONAR OBJETOS PRA  
PREENCHER O VAZIO?  
EIS A SOLUÇÃO!

```
# NOVO: Desenhar um retângulo vermelho
# pygame.draw.rect(tela, cor, (x, y, Largura, altura))
pygame.draw.rect(screen, (255, 0, 0), (175, 125, 50, 50))
```

PYGAME.DRAW.RECT(TELA, (R,G,B), (X,Y,Z,W))

POSIÇÕES NA TELA: X,Y  
DIMENSÕES DO OBJETO: Z,W

# EXIBIR O RETÂNGULO VERMELHO

```
pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Desenhando Formas")
clock = pygame.time.Clock()

running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Limpa a tela
    screen.fill((50, 50, 50)) # Cinza escuro

    # NOVO: Desenhar um retângulo vermelho
    # pygame.draw.rect(tela, cor, (x, y, largura, altura))
    pygame.draw.rect(screen, (255, 0, 0), (175, 125, 50, 50))

    pygame.display.flip()

pygame.quit()
sys.exit()
```

PODEMOS CRIAR 2 VARIÁVEIS:

QUADRADO\_X E QUADRADO\_Y

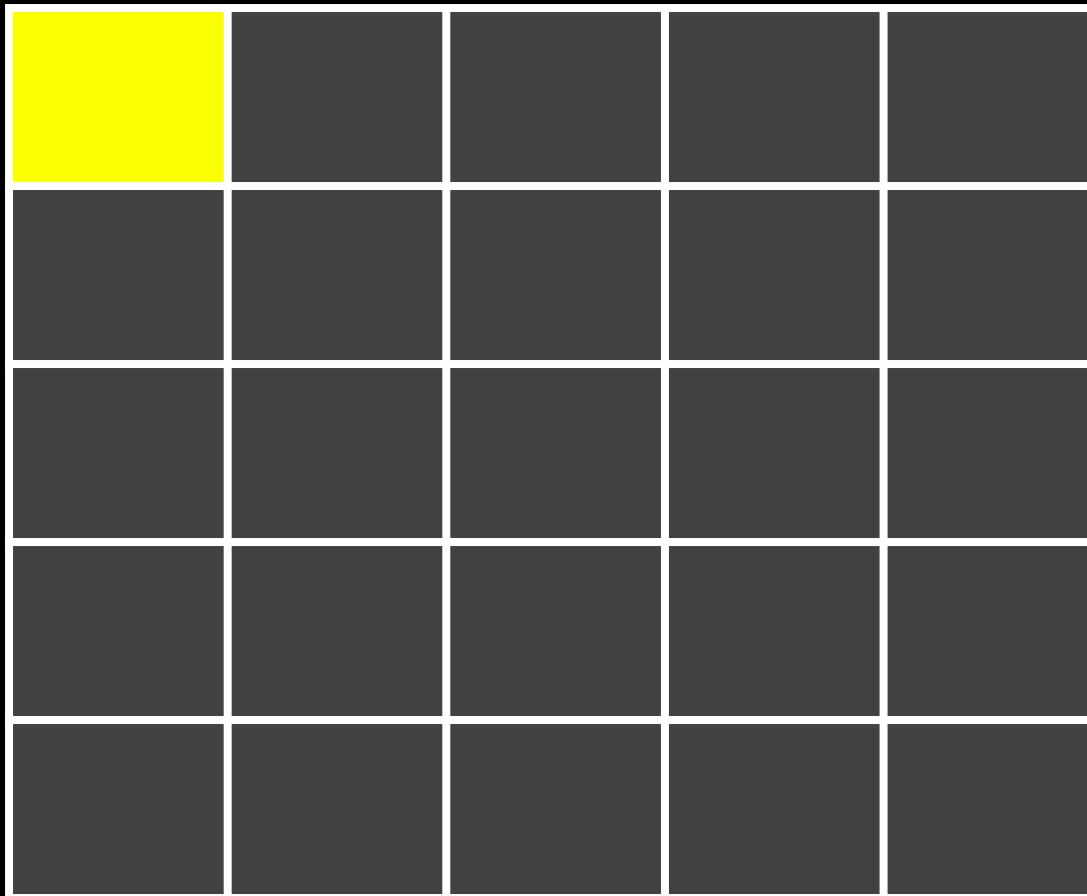
DEFINIR ELAS COMO POSIÇÃO DO QUADRADO!

```
# NOVO: Desenha o quadrado na posição atualizada
pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, 50, 50))
```

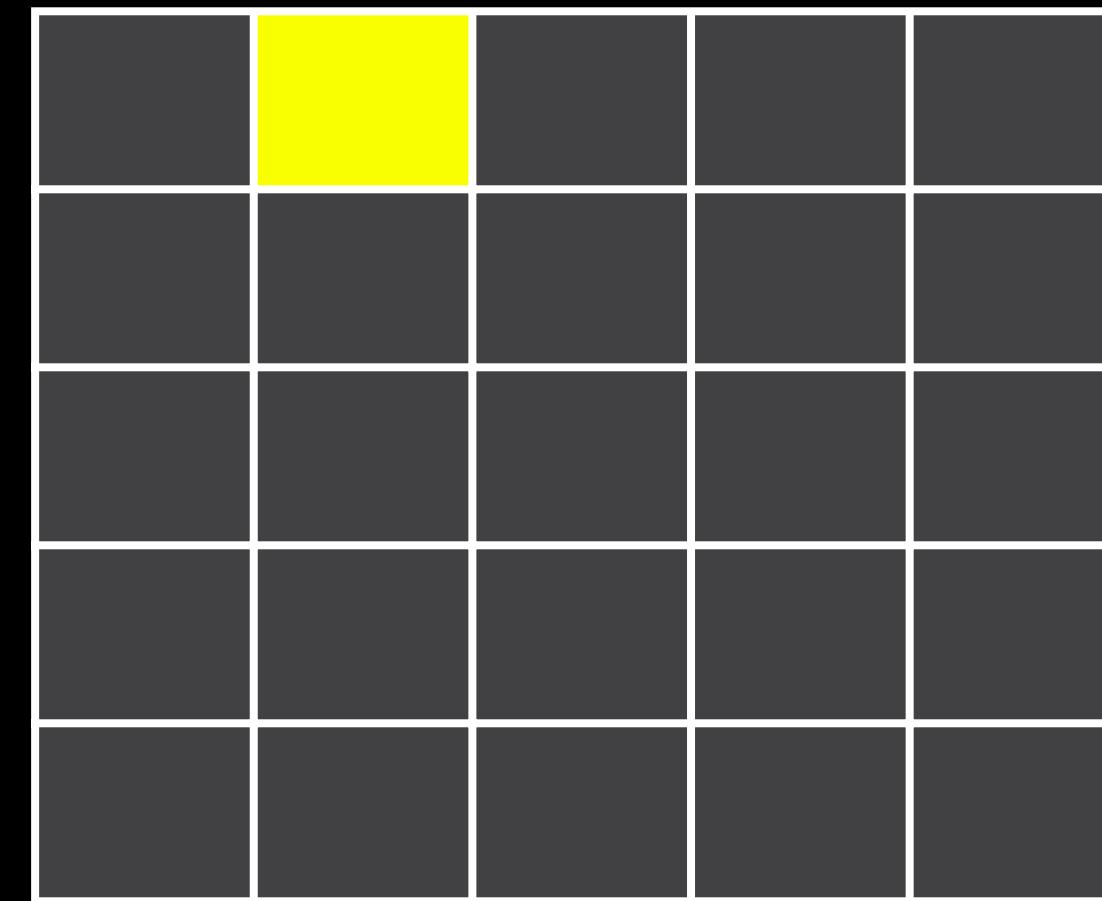
SE MUDARMOS AS VARIÁVEIS X, E Y, O QUADRADO MUDARA DE LUGAR NA TELA!

# MOVIMENTAÇÃO POR FRAMES

COMO A VELOCIDADE PODE SER  
"SIMULADA" EM UM AMBIENTE 2D?



FRAME ( $N$ ) + 1 =



POSICAO = 0,0

0( $X+1$ ), V)

# EXIBIR O RETÂNGULO VERMELHO

CRIAMOS A VARIÁVEL VELOCIDADE!

```
# NOVO: Variáveis para posição do quadrado  
quadrado_x = 175  
quadrado_y = 125  
velocidade = 5 # Pixels por frame
```

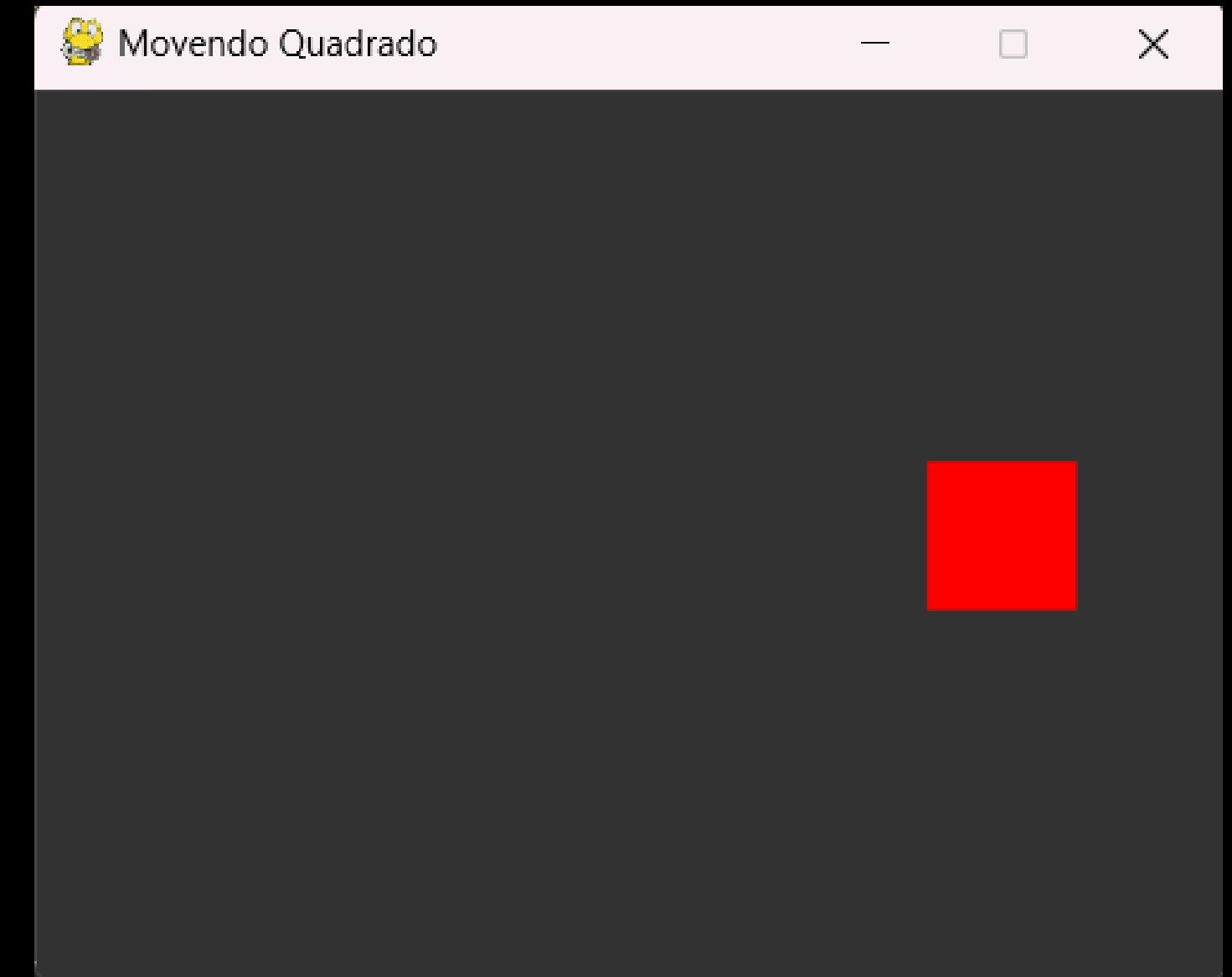
## DESAFIOS

```
# NOVO: Verifica quais teclas estão pressionadas AGORA  
teclas = pygame.key.get_pressed()  
  
# NOVO: Move o quadrado baseado nas teclas  
if teclas[pygame.K_LEFT]:  
    pass  
if teclas[pygame.K_RIGHT]:  
    pass  
if teclas[pygame.K_UP]:  
    pass  
if teclas[pygame.K_DOWN]:  
    pass  
  
# Limpa a tela  
screen.fill((50, 50, 50))  
  
# NOVO: Desenha o quadrado na posição atualizada  
pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, 50, 50))  
pygame.display.flip()
```

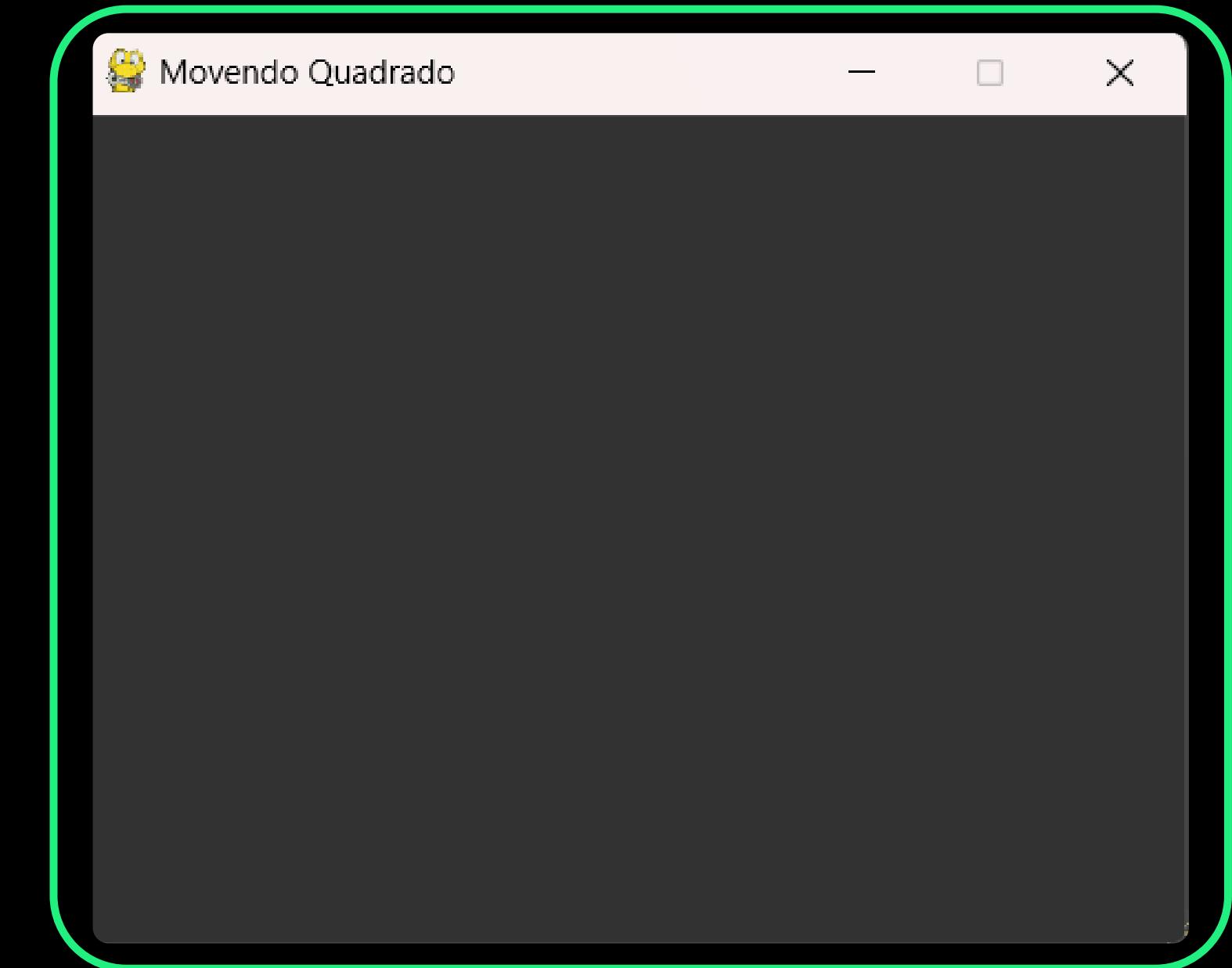
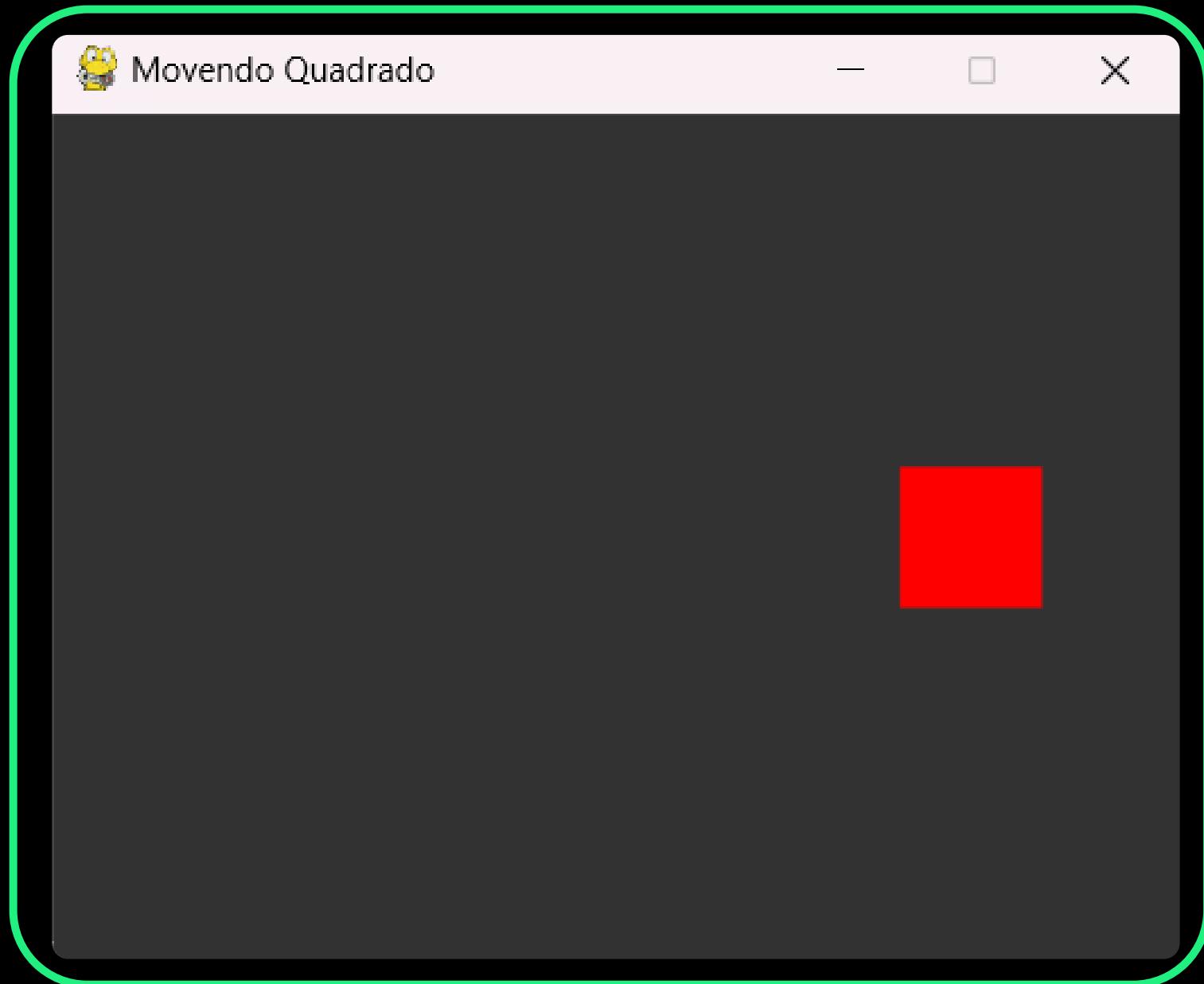
COMO PODEMOS  
MOVIMENTAR O  
QUADRADO PELA TELA ?

# RESULTADO:

```
if teclas[pygame.K_LEFT]:  
    quadrado_x -= velocidade  
if teclas[pygame.K_RIGHT]:  
    quadrado_x += velocidade  
if teclas[pygame.K_UP]:  
    quadrado_y -= velocidade  
if teclas[pygame.K_DOWN]:  
    quadrado_y += velocidade
```



# TEMOS UM PROBLEMA!

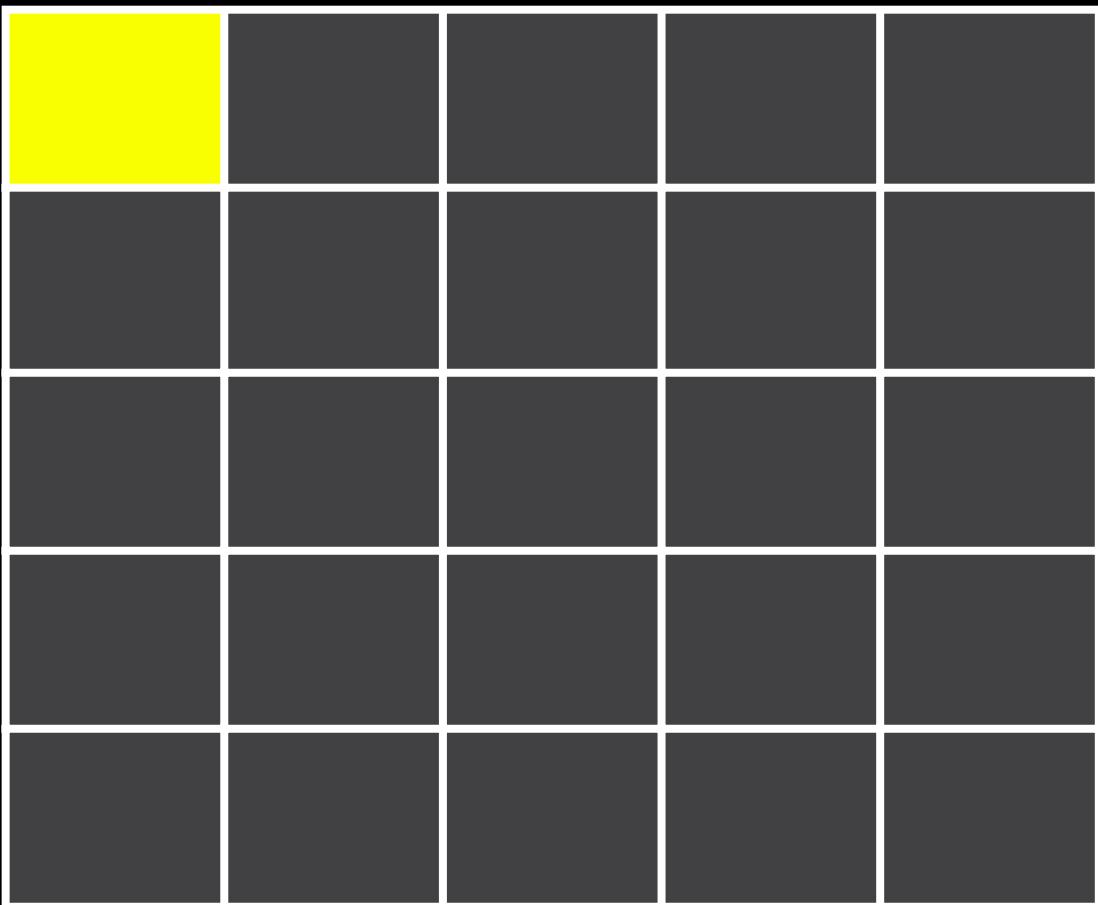


SE ANDARMOS PRA FORA, O  
NOSSO QUADRADO VAI  
EMBORA!

# COMO PODEMOS CONSESTAR?

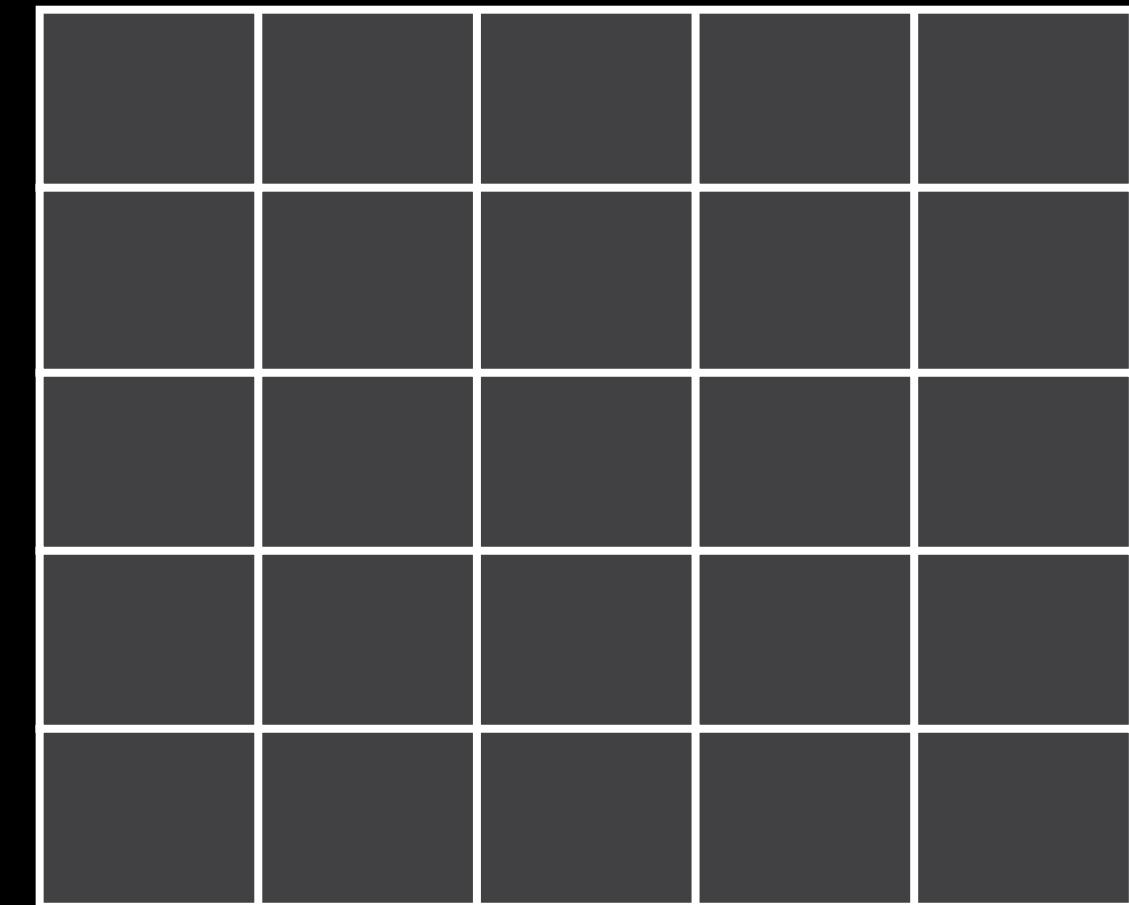
## NOVAMENTE IMAGINANDO O PLANO CARTESIANO 2D

M POR N



((H), V)

M POR N



((H-1), V)

FRAME (N) + 1 =

# COMO PODEMOS CONSELTAR?

NOVAMENTE IMAGINANDO O  
PLANO CARTERSIANO 2D

M POR N


$(x-1), y)$

SE TEMOS X, Y E UMA MATRIZ M  
POR N

X NÃO PODE SER MAIOR QUE M  
Y NÃO PODE SER MAIOR QUE N

COMO RESULTADO, CRIAMOS  
UMA PAREDE INVISIVEL !

# DESAFIO!

## NÃO DEIXE O QUADRADO SAIR PRA FORA DA TELA!

SENDO TAMANHO = 50,  
(TAMANHO DO QUADRADO)

```
# NOVO: Limitar posição dentro da tela
# Não pode ser menor que 0
if quadrado_x < 0:
    pass
# Não pode ser maior que largura da tela - tamanho do quadrado
if quadrado_x > 400 - tamanho:
    pass
# Mesma lógica para Y
if quadrado_y < 0:
    pass
if quadrado_y > 300 - tamanho:
    pass
```

SE TEMOS X,Y E UMA MATRIZ M  
POR N

Y NÃO PODE SER MAIOR QUE N  
X NÃO PODE SER MAIOR QUE M

# RESULTADO:

```
# NOVO: Limitar posição dentro da tela
# Não pode ser menor que 0
if quadrado_x < 0:
    quadrado_x = 0
# Não pode ser maior que Largura da tela - tamanho do quadrado
if quadrado_x > 400 - tamanho:
    quadrado_x = 400 - tamanho
# Mesma lógica para Y
if quadrado_y < 0:
    quadrado_y = 0
if quadrado_y > 300 - tamanho:
    quadrado_y = 300 - tamanho

screen.fill((50, 50, 50))
pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho, tamanho))
pygame.display.flip()
```

# MECÂNICAS CORE

SÃO OS SISTEMAS DE GAMEPLAY FUNDAMENTAIS QUE DEFINEM A ESSÊNCIA DE COMO O JOGO É JOGADO.



SUBWAY SURFERS

CHEGAR O MAIS  
LONGE POSSIVEL!

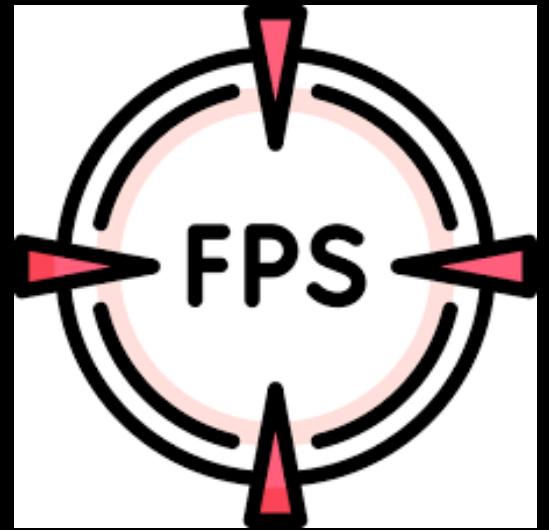


ANGRY BIRDS

DERROTAR TODOS  
OS PORQUINHOS!

# MECÂNICAS CORE

DEFINEM O GÊNERO: FREQUENTEMENTE IDENTIFICAM  
O TIPO DE JOGO.



**FPS**

ATIRAR,  
MOVIMENTAR-SE



**SURVIVAL**

SOBREVIVER  
(OPCIONAL)



**RPG**

EVOLUIR, VIVER A  
HISTÓRIA

# MECÂNICAS CORE



NEM SEMPRE PRECISA SER ALGO MUITO COMPLEXO!

# MECÂNICAS CORE



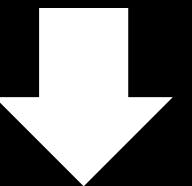
PRECISA INSTIGAR O JOGADOR!

# MECÂNICAS CORE

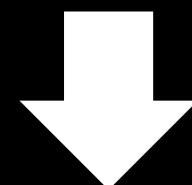


O EXEMPLO DE DARK SOULS!

FALHAR FAZ PARTE DA  
MECÂNICA DO JOGO



O JOGADOR É RECOMPENSADO  
POR CONTINUAR TENTANDO



O JOGADOR SE SENTE FELIZ

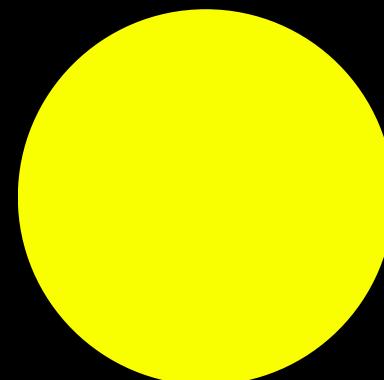
# VAMOS CRIAR UMA MECÂNICA!

COMEÇANDO SIMPLES, VAMOS FAZER UMA  
MOEDA PARA O NOSSO QUADRADO!

VAMOS DEFINIR AS VARIÁVEIS  
PARA UM CÍRCULO:

```
CIRCULO_X = 100  
CIRCULO_Y = 100  
CIRCULO_RAIO = 20
```

```
PYGAME.DRAW.CIRCLE(SCREEN, (255, 255, 0), (CIRCULO_X, CIRCULO_Y), CIRCULO_RAIO)
```



# VAMOS CRIAR UMA MECÂNICA!

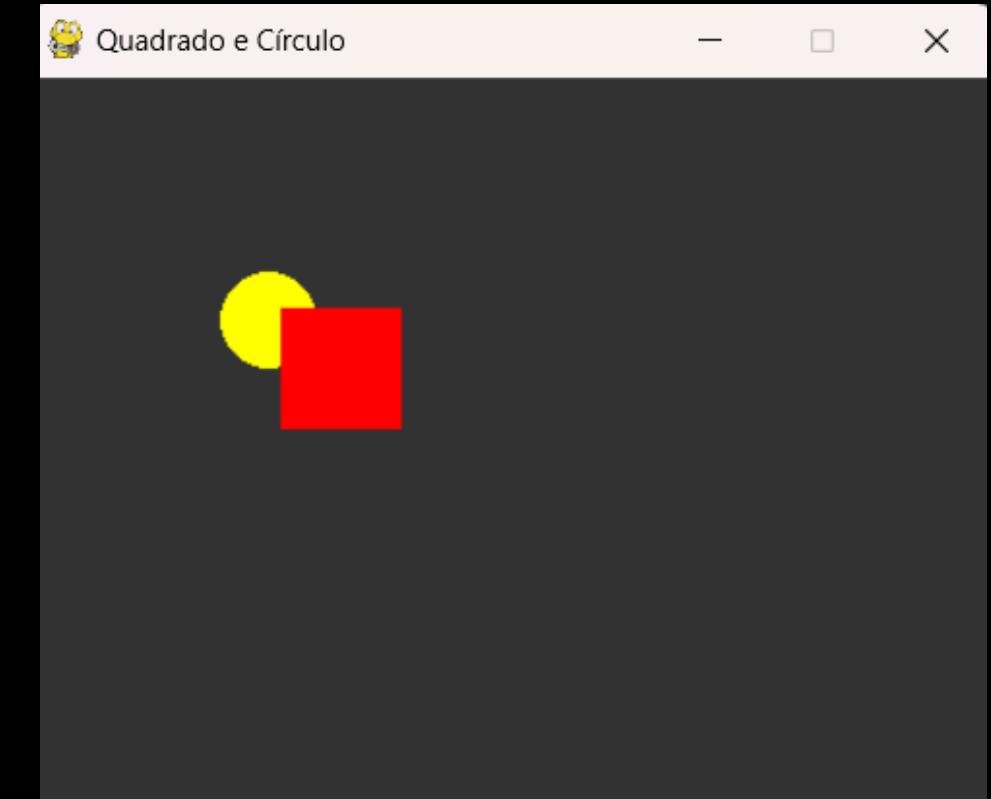
## COMEÇANDO SIMPLES, VAMOS FAZER UMA MOEDA PARA O NOSSO QUADRADO!

```
# Desenhar tudo
screen.fill((50, 50, 50))

# NOVO: Desenhar círculo
# pygame.draw.circle(tela, cor, (centro_x, centro_y), raio)
pygame.draw.circle(screen, (255, 255, 0), (circulo_x, circulo_y), circulo_raio)

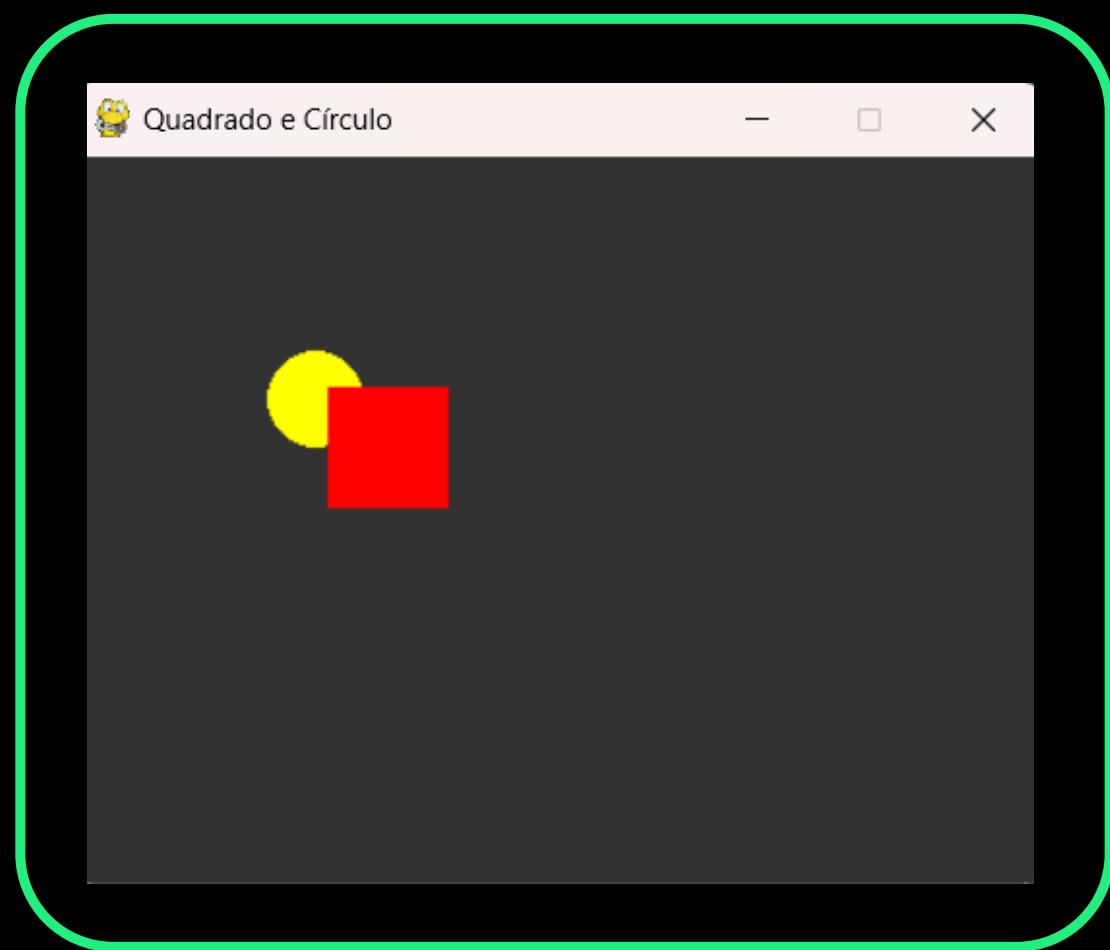
# Desenhar quadrado (por cima)
pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho, tamanho))

pygame.display.flip()
```



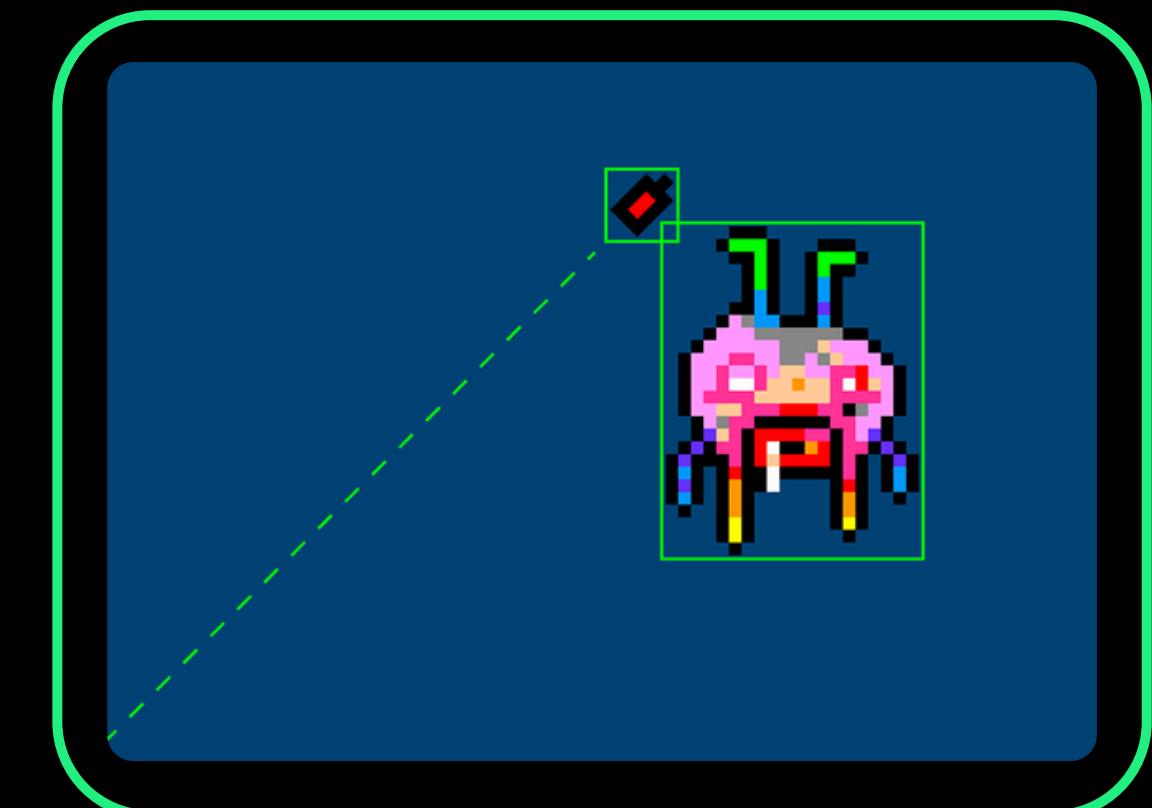
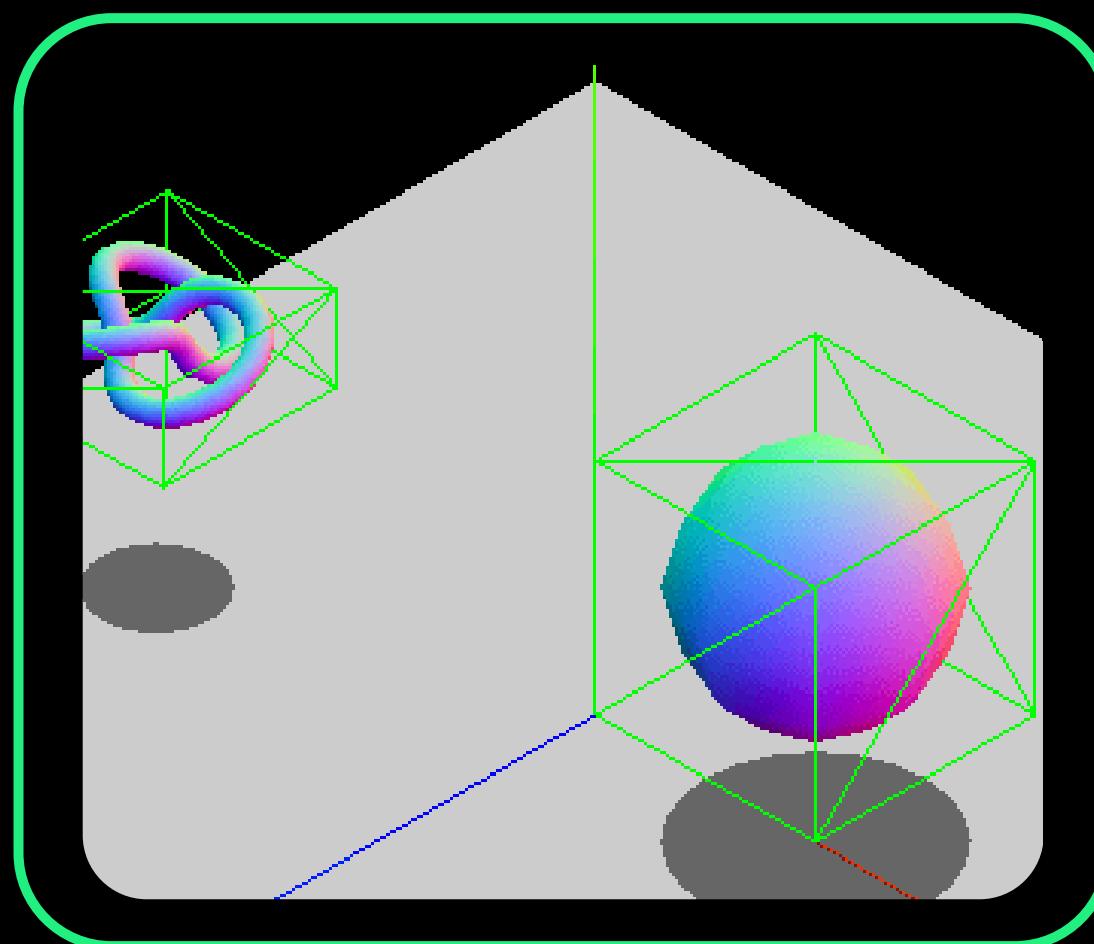
# EVENTOS DE COLISÃO!

COMO LIDAMOS COM COLISÃO E CRIAMOS  
OBJETOS SOLIDOS?



# EVENTOS DE COLISÃO!

## SOLUÇÕES DO ESTADO DA ARTE:

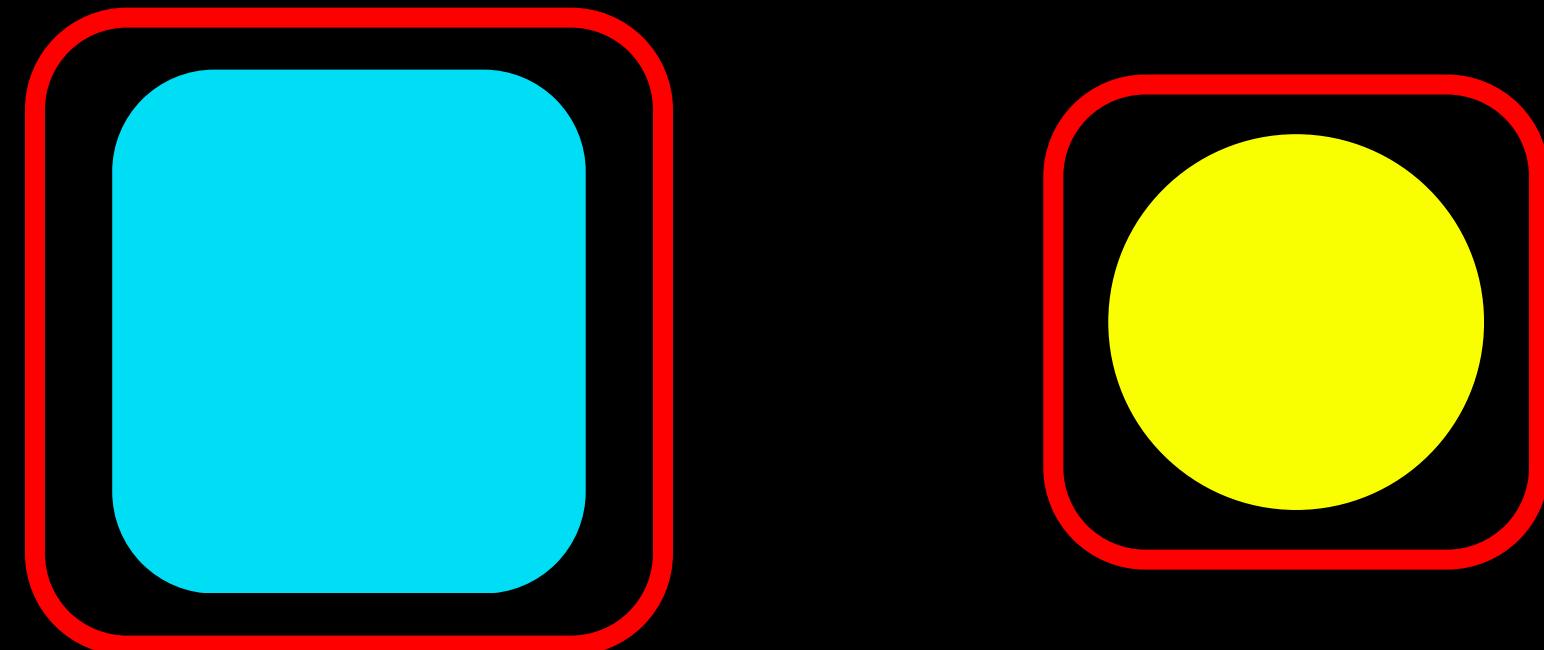


# EVENTOS DE COLISÃO!

## VAMOS IMPLEMENTAR!

Novo conceito: pygame.Rect

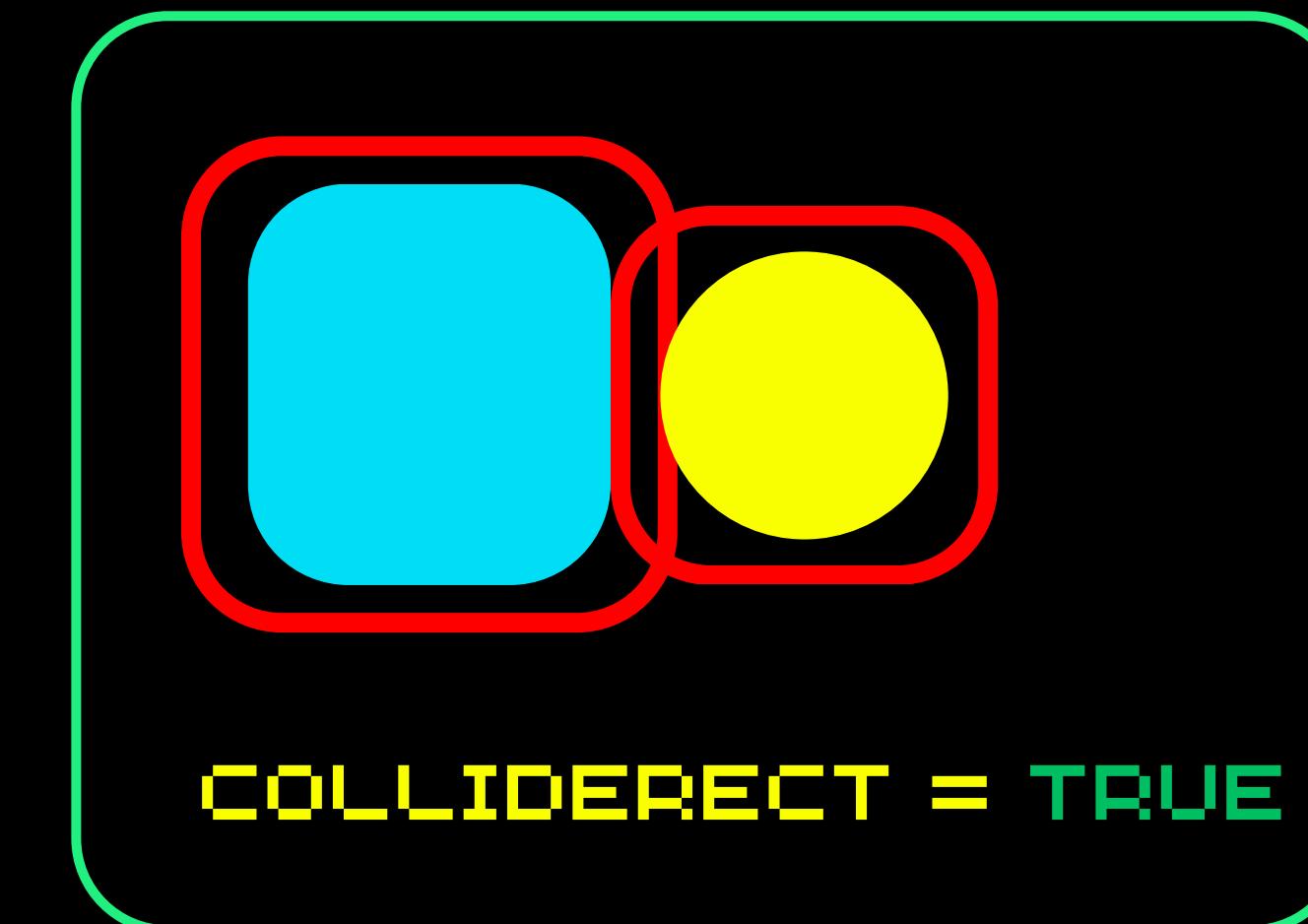
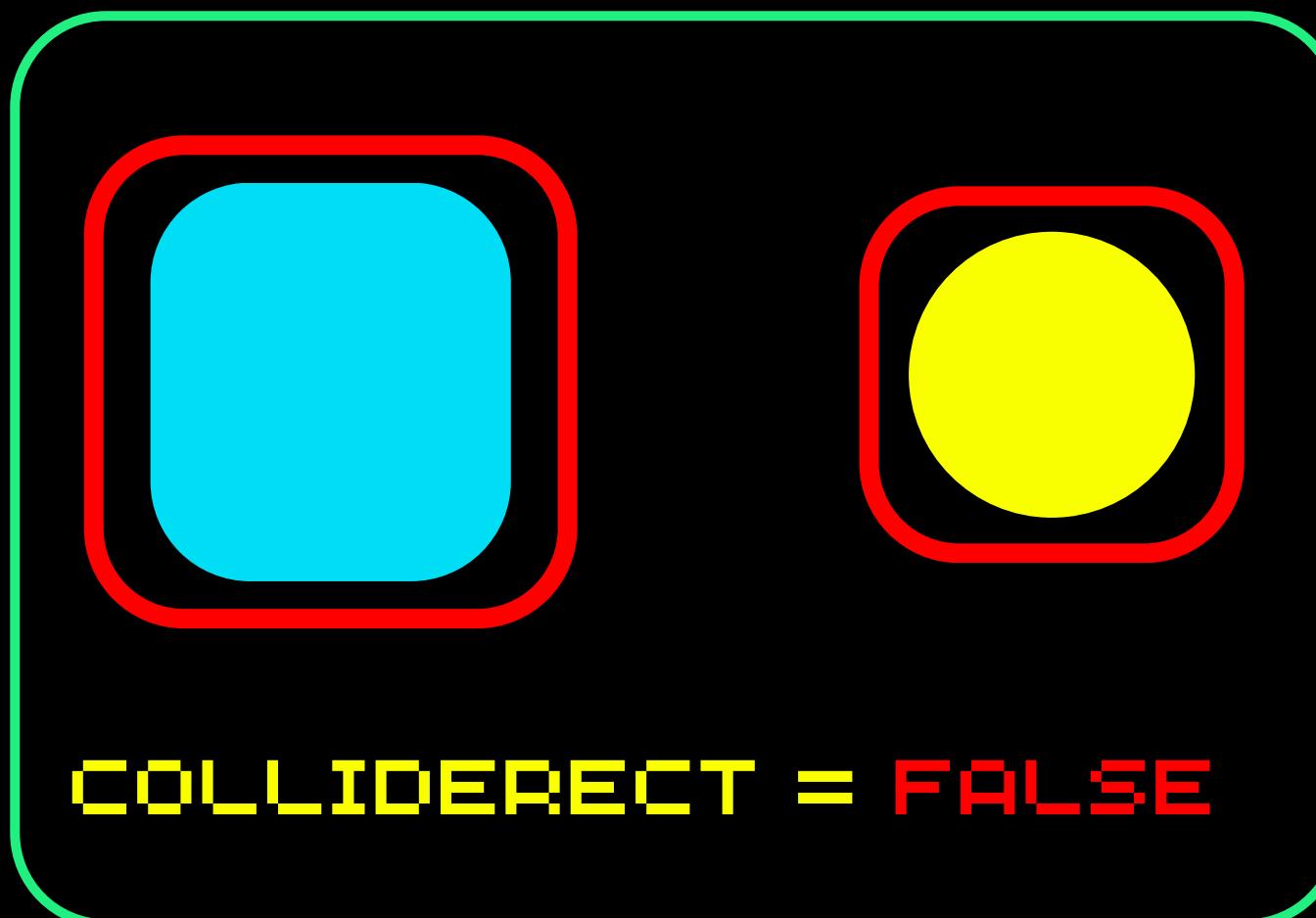
```
# NOVO: Criar retângulos para colisão
# Retângulo do quadrado
rect_quadrado = pygame.Rect(quadrado_x, quadrado_y, tamanho, tamanho)
# Retângulo ao redor do círculo (aproximação)
rect_circulo = pygame.Rect(circulo_x - circulo_raio,
                            circulo_y - circulo_raio,
                            circulo_raio * 2,
                            circulo_raio * 2)
```



# EVENTOS DE COLISÃO!

## VAMOS IMPLEMENTAR!

Novo conceito: (rect).colliderect()



# EVENTOS DE COLISÃO!

## VAMOS IMPLEMENTAR!

MAS COMO SABEMOS O ESTADO DA MOEDA?

CRIAMOS UMA NOVA VARIÁVEL:

```
MOEDA_COLETADA = FALSE # NOVO: FLAG PARA SABER SE COLETOU
```

```
# Círculo (moeda)
circulo_x = 100
circulo_y = 100
circulo_raio = 20
moeda_coletada = False # NOVO: Flag para saber se coletou
```

# EVENTOS DE COLISÃO! DESAFIO!

## VAMOS FAZER A MECÂNICA DE COLETAR A MOEDA!

```
# NOVO: Criar retângulos para colisão
# Retângulo do quadrado
rect_quadrado = pygame.Rect(quadrado_x, quadrado_y, tamanho, tamanho)
# Retângulo ao redor do círculo (aproximação)
rect_circulo = pygame.Rect(circulo_x - circulo_raio,
                            circulo_y - circulo_raio,
                            circulo_raio * 2,
                            circulo_raio * 2)

# NOVO: Verificar colisão
if colisao ?:

    # Desenhar tudo
    screen.fill((50, 50, 50))

    # NOVO: Só desenha o círculo se NÃO foi coletado
    if not moeda_coletada:
        pass

    pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho, tamanho))

    pygame.display.flip()
```

# RESULTADO:

```
# NOVO: Criar retângulos para colisão
# Retângulo do quadrado
rect_quadrado = pygame.Rect(quadrado_x, quadrado_y, tamanho, tamanho)
# Retângulo ao redor do círculo (aproximação)
rect_circulo = pygame.Rect(circulo_x - circulo_raio,
                            circulo_y - circulo_raio,
                            circulo_raio * 2,
                            circulo_raio * 2)

# NOVO: Verificar colisão
if rect_quadrado.colliderect(rect_circulo) and not moeda_coletada:
    moeda_coletada = True # Marca como coletada
    print("Moeda coletada!") # Mensagem no terminal

# Desenhar tudo
screen.fill((50, 50, 50))

# NOVO: Só desenha o círculo se NÃO foi coletado
if not moeda_coletada:
    pygame.draw.circle(screen, (255, 255, 0), (circulo_x, circulo_y), circulo_raio)

pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho, tamanho))

pygame.display.flip()
```

# DESAFIO FINAL

## GAMEPLAY LOOP



VOCÊ CONSEGUE ALEATORIEZAR A POSIÇÃO DA MOEDA?

# RESULTADO:

```
# Função para gerar posição aleatória para a moeda
def gerar_posicao_moeda():
    """
    Gera uma posição aleatória dentro da tela,
    respeitando as bordas para a moeda não aparecer cortada
    """
    # randint gera um número inteiro aleatório entre min e max (inclusive)
    x = random.randint(circulo_raio, 400 - circulo_raio)
    y = random.randint(circulo_raio, 300 - circulo_raio)
    return x, y

# Verificar colisão com a moeda
if rect_quadrado.colliderect(rect_circulo):
    pontos += 1 # Aumenta pontuação
    print(f"Moeda coletada! Total: {pontos}")

# Gera nova posição aleatória para a moeda
circulo_x, circulo_y = gerar_posicao_moeda()
```

**PREMIAÇÃO!**

# PREMIAÇÃO PARA O VENCEDOR DOS DESAFIOS!



Clique para ver a visualização completa

Mouse Gamer Redragon Impact Preto  
RGB M908

Visite a loja Redragon  
4,5  534 avaliações de clientes

R\$ 262<sup>99</sup>

Em até 6x R\$ 43,84 sem juros [Ver parcelas disponíveis](#)

[Pagamentos e Segurança](#) [Política de devolução](#)

Marca	Redragon
Cor	Preto
Tecnologia de conectividade	USB
Características especiais	Luzes de LED
Tecnologia de detecção de movimento	Óptico

**Sobre este item**

- Poderoso Sensor Pixart PMW 3327 de 12400 DPI com 5 modos de DPI (500/1000/2000/3000/6200 DPI) (30G ACC)
- Polling Rate de 1000hz (Tempo de Resposta Ajustável via Software de 1/2/4/8ms)
- Retroiluminação LED RGB Ajustável
- Cabo trançado de alta Qualidade para maior durabilidade
- Ajuste de Peso para uma experiência única personalizada

# OBRIGADO!



@jader.louis



@Prism411

jaderlouis@proton.me

**Prism411/VIII-Semana-da-Computacao-...**

Este repositório contém todos os exercícios e desafios desenvolvidos durante a VIII Semana da Computação, no curso de Introdução ao...

Contributor 1 Issues 0 Stars 0 Forks 0

---

**Prism411/VIII-Semana-da-Computacao-GAMEDEV-COURSE: Este repositório contém todos os exercícios e...**

Este repositório contém todos os exercícios e desafios desenvolvidos durante a VIII Semana da Computação, no curso de Introdução ao Game Dev ministrado por Jáder Louis de Souza Gonçalves. - Prism41...

[GitHub](#)

FIM